Pós-Graduação em Ciência da Computação

EDSON ALVES DA SILVA

Uma Linguagem Visual para Diagramar Consultas SQL

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2020

EDSON ALVES DA SILVA

Uma Linguagem Visual para Diagramar Consultas SQL

Tese de Doutorado apresentada ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Doutor em Ciência da Computação.

Área de Concentração: Banco de dados
Orientador: Robson do Nascimento Fidalgo

Recife
2020

Catalogação na fonte
Bibliotecária: Mônica Uchôa, CRB4-1010

S586l Silva, Edson Alves da.
Uma linguagem visual para diagramar consultas SQL / Edson Alves da Silva. – 2020.
136 f.: il., fig., tab.

Orientador: Robson do Nascimento Fidalgo.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn. Programa de Pós-graduação em Ciência da Computação. Recife, 2020.
Inclui referências e apêndices.

1. Banco de dados. 2. SQL. I. Fidalgo, Robson do Nascimento (Orientador). II. Título.

681.3 CDD (23. ed.) UFPE- CCEN 2021 - 191

Edson Alves da Silva

“Uma Linguagem Visual para Diagramar Consultas SQL”

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Aprovado em: 13/03/2020.

_____

Orientador: Prof. Dr. Robson do Nascimento Fidalgo

BANCA EXAMINADORA

_____

Profa. Dra. Ana Carolina Brandão Salgado
Centro de Informática / UFPE

_____

Prof. Dr. Vinicius Cardoso Garcia
Centro de Informática / UFPE

_____

Prof. Dr. Sergio Lifschitz
Departamento de Informática / PUC/RJ

_____

Prof. Dr. Claudio de Souza Baptista
Centro de Engenharia Elétrica e Informática, Sistemas e Computação/UFCG

_____

Prof. Dr. Altigran Soares da Silva
Instituto de Computação / UFAM

Decido este trabalho a minha família que foi porto seguro perante as dificuldades durante este percurso.

AGRADECIMENTOS

Agradeço a meu orientador, Robson, que me acolheu e orientou durante todos estes anos. Além dos diversos papéis que lhe cabem oficialmente, sendo uma orientador dedicado, interessado e competente, ele me ajudou muito dentro e fora do doutorado.

Obrigado também aos membros da banca examinadora, Ana Carolina Salgado, Vinicius Cardoso Garcia, Sergio Lifschitz, Cláudio de Souza Baptista e Altigran Soares da Silva, que com seus valiosos comentários durante a defesa ajudaram a enriquecer muito este trabalho.

Agradeço também às instituições que fizeram com que esta trajetória fosse possível. Estas incluem o CIn-UFPE, por ter sido a minha casa durante estes anos. Também agradeço à FACEPE, que me ajudou financeiramente. Espero ter correspondido e continuar correspondendo à altura este investimento em mim realizado.

Agradeço também aos amigos que encontrei pelo caminho, pois foram muito importantes durante essa trajetória: Natália, Thiago, Robério, Gênesis e Augusto.

Por fim, agradeço à minha amada esposa Caline, por todo o amor e carinho que me fazem lembrar que a vida não podia ser melhor. E também pelo apoio, força e compreensão nos (muitos) momentos difíceis. Também agradeço a meus pais Maria e José, meus irmãos Eduardo, Ednaldo e Edjane, minha cunhada Carolaine, e meus sogros Paulo e Iraildes.

E obrigado DEUS, pela minha vida.

RESUMO

A linguagem *Structured Query Language* (SQL) é amplamente usada para acessar bancos de dados relacionais e não relacionais. Nos bancos de dados relacionais, SQL é a forma padrão de acesso. Nos bancos de dados não relacionais, SQL está se tornando cada vez mais disponível e se consolidando como uma interface de acesso para consultar dados em ambientes de *cluster* (e.g., Apache Hive e Spark SQL). Apesar de sua sintaxe declarativa, a especificação de consultas SQL não é uma tarefa trivial (mesmo para especialistas), porque algumas consultas exigem construções complexas (i.e., subconsultas, junções, operações de conjunto, expressões condicionais, restrições de agrupamento e recursões). As linguagens visuais de consulta (*Visual Query Language* - VQL) são uma alternativa que visam reduzir essa complexidade. No entanto, embora várias VQL tenham sido propostas, uma revisão do estado da arte verificou que essas VQL não são amplamente utilizadas na prática, pois não abrangem várias construções complexas e não possuem ferramentas *Computer Aided Software Engineering* (CASE) disponíveis para seus usuários finais, comprometendo sua expressividade e disponibilidade. Visando superar essas limitações, o objetivo desta tese é especificar a sintaxe concreta (i.e., notação gráfica) e a sintaxe abstrata (i.e., metamodelo) de uma VQL denominada *Diagrammatic Structured Query Language* (DSQL), que considere, ao mesmo tempo, todas as construções complexas acima e seja tão compreensível e eficiente quanto SQL sem aumento do esforço. Esta tese inicia-se com uma revisão sistemática da literatura, a qual visa entender porque as VQL não são amplamente utilizadas na prática. Além disso, o paradigma *Model-Driven Development* (MDD) é usado como arcabouço teórico e tecnológico para a especificação de DSQL. Por fim, para avaliar a expressividade de DSQL é apresentado um experimento para comparar a precisão, o tempo e o esforço para compreender consultas complexas usando DSQL e SQL. Como resultados têm-se: 1) a revisão sistemática sobre o estado da arte de VQL para SQL; 2) a especificação da sintaxe concreta e da sintaxe abstrata de DSQL; e 3) os resultados do experimento, os quais indicam que não há diferença significativa entre as duas linguagens, mas DSQL é mais rápida de compreender do que SQL. A avaliação dá indícios de que DSQL tem potencial para ser tão compreensível e eficiente quanto SQL sem aumento do esforço e que DSQL avança o estado da arte das VQL, pois mitiga as limitações das propostas relacionadas. Portanto, DSQL pode ser uma alternativa para usuários que preferem trabalhar com uma notação visual em vez de sintaxe textual.

Palavras-chaves: SQL. Consulta Complexa. Linguagem Visual de Consulta.

ABSTRACT

Structured Query Language (SQL) is a widely-used language for accessing both relational and non-relational databases. In relational databases, SQL is the standard form of access. In non-relational databases, SQL is becoming increasingly available and consolidating itself as an access interface for querying data in cluster environments (e.g., Apache Hive and Spark SQL). Despite its declarative syntax, the specification of SQL queries is not a trivial task (even for experts), because some queries demand complex constructs (i.e., subqueries, joins, set operations, conditional expressions, grouping restrictions, and recursions). Visual Query Languages (VQL) are an alternative that aims to reduce this complexity. However, although several VQL have been proposed, a review of the state of the art verified that these VQL are not widely used in practice because they do not cover many complex constructs and do not have Computer Aided Software Engineering (CASE) tools available to their end users, compromising their expressiveness and availability. To overcome these limitations, the purpose of this thesis is to specify the concrete syntax (i.e., graphical notation) and abstract syntax (i.e., metamodel) of a VQL, called Diagrammatic Structured Query Language (DSQL), which considers all the above complex constructs at the same time. This thesis begins with a systematic literature review, which aims to understand why VQL are not widely used in practice. In addition, the Model-Driven Development (MDD) paradigm is used as a theoretical and technological framework for DSQL specification. Finally, to evaluate the expressiveness of DSQL, an experiment is presented to compare the accuracy, time and effort to understand complex queries using DSQL and SQL. As a result we have: 1) the systematic review of the state of the art from VQL to SQL; 2) the specification of DSQL concrete syntax and abstract syntax; and 3) the results of the experiment, which indicate that there is no significant difference between the two languages, but DSQL is faster to understand than SQL. The experimental evaluation has shown that DSQL has the potential to be as understandable and efficient as SQL without increasing effort and that DSQL advances the state of the art of VQL by mitigating the limitations of related proposals. Therefore, DSQL may be an alternative for users who prefer to work with visual notation rather than textual syntax.

Keywords: SQL. Complex Query. Visual Query Language.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CASE	*Computer Aided Software Engineering*
CTE	*Common Table Expression*
DML	*Data Manipulation Language*
DSML	*Domain-Specific Modeling Language*
DSQL	*Diagrammatic Structured Query Language*
EMF	*Eclipse Modeling Framework*
EMOF	*Essential Meta Object Facility*
EVL	*Epsilon Validation Language*
GMP	*Graphical Modeling Project*
MDD	*Model-Driven Development*
MOF	*Meta Object Facility*
OCL	*Object Constraint Language*
RSL	Revisão Sistemática da Literatura
SQL	*Structured Query Language*
VQL	*Visual Query Language*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo apresenta-se esta tese, destacando o seu contexto, motivação e objetivos, bem como a delimitação do escopo e metodologia. Por fim, apresenta-se a estrutura dos demais capítulos.

1.1 CONTEXTO

A *Structured Query Language* (SQL) (CHAMBERLIN; BOYCE, 1974) é uma linguagem declarativa definida como o padrão para acessar bancos de dados relacionais. Apesar do popularização do uso de *MapReduce* (DEAN; GHEMAWAT, 2004) para processamento de dados nos banco de dados não relacionais, a SQL continua sendo a *lingua franca* para processamento de dados, visto que estes bancos geralmente envolvem alguma forma de consulta baseada em SQL (e.g., Apache Hive (THUSOO et al., 2009) e Spark SQL (ARMBRUST et al., 2015)) e que a SQL está disponível nos banco de dados newSQL mais atuais (e.g., CockroachDB (COCKROACH, 2020) e Spanner (BACON et al., 2017)). Por esse motivo, SQL permanece uma tecnologia importante para a academia e a indústria. Na academia, os currículos mais importantes de ciência da computação (ACM/IEEE, 2013; ACM/IEEE, 2017) recomendam o ensino de SQL. Por outro lado, na indústria (STACKOVERFLOW, 2018), SQL se tornou uma das linguagens mais populares entre desenvolvedores *Web*, desenvolvedores *desktops*, *sysadmins*/*DevOps* e cientistas de dados devido à sua maturidade e facilidade de entendimento e uso.

Figura 1 – Exemplos de consultas SQL com construções complexas

```
SELECT AVG(qty)
FROM
   (SELECT orderId, COUNT(*) as qty
    FROM orderItem
    GROUP BY orderId) as qi;
```

(a) Qual é o número médio de itens por pedido?

```
SELECT model
FROM product
GROUP BY model
HAVING AVG(price) >
   (SELECT AVG(p.price)
    FROM product p)
```

(b) Quais são os modelos dos produtos com preço médio superior ao preço médio geral?

Embora a sintaxe básica de SQL seja fácil de entender, especificar consultas que usam construções complexas é uma atividade não trivial. Isto é, a consulta tem potencial para demandar mais esforço intelectual e técnico quando envolve ao menos uma das seguintes construções: 1) subconsulta (correlacionada ou não correlacionada) (CZEJDO et al., 1993; KAWASH, 2004; ZHANG et al., 1999; CATARCI; SANTUCCI, 1995); 2) junção (cruzada, interna ou externa) (CATARCI; SANTUCCI, 1995); 3) operação de conjunto (união, exceto ou interseção) (EL-MAHGARY; SOISALON-SOININEN, 2015); 4) expressão condicional (cláusulas *Case-When*) (GRYZ et al., 2008); 5) restrição de agrupamento (cláusula *Having*) (ZHANG et al., 1999); e 6) recursão (cláusula *With*) (EL-MAHGARY; SOISALON-SOININEN, 2015). Por exemplo, na Figura 1 apresentam-se duas consultas SQL com construções complexas. Na Figura 1a mostra-se uma

subconsulta não correlacionada na cláusula *From* e na Figura 1b exibe-se uma restrição de agrupamento e uma subconsulta não correlacionada na cláusula *Having*.

Considerando que consultas SQL podem envolver construções complexas, uma linguagem visual de consulta (*Visual Query Language* - VQL) (CATARCI et al., 1997) é uma alternativa que pode reduzir essa complexidade (cf. Seção 2.3). Diferentemente das representações textuais, as abstrações visuais de uma VQL permitem que os usuários se concentrem nos aspectos semânticos ao invés dos aspectos sintáticos das consultas (CATARCI; SANTUCCI, 1995; HVORECKý; DRLíK; MUNK, 2010). Isso favorece o desenvolvimento do pensamento crítico e criativo, uma vez que as abstrações visuais das VQL criam condições que melhoram o pensamento analítico e as habilidades de resolução de problemas. Além disso, as abstrações das VQL permitem que as informações sejam interpretadas mais rapidamente do que a notação textual de SQL, pois podem explorar a capacidade do sistema visual humano de processar informações em paralelo (MOODY, 2009; WARE, 2004).

Uma VQL pode ser desenvolvida com base no paradigma *Model-Driven Development* (MDD) (BRAMBILLA; CABOT; WIMMER, 2017; KELLY; TOLVANEN, 2008), cujo maior benefício é a especificação de modelos executáveis. Isto é, diagramas que abstraem a complexidade da sintaxe de uma linguagem computacional e são usados por ferramentas *Computer Aided Software Engineering* (CASE) para gerar, automaticamente, código executável ou interpretável (cf. Seção 2.2). O uso de MDD favorece o aumento de produtividade e ajuda a reduzir o número de erros, a quantidade de tempo necessário e os custos de desenvolvimento. Uma VQL desenvolvida com base no paradigma MDD é uma linguagem de modelagem específica de domínio (*Domain-Specific Modeling Language* - DSML) (BRAMBILLA; CABOT; WIMMER, 2017), pois esta é dedicada ao domínio do problema do usuário, uma vez que usa notações e abstrações específicas que permitem aumentar a compreensão e a produtividade, bem como evitar ou alertar erros em um dado domínio. A proposta de uma DSML consiste na especificação da sua sintaxe abstrata (i.e., metamodelo ou gramática), sintaxe concreta (notação gráfica ou representação visual) e semântica (significado, incluído regras de boa formação) (ACHILLEOS; GEORGALAS; YANG, 2007; KELLY; TOLVANEN, 2008; BRAMBILLA; CABOT; WIMMER, 2017).

1.2 MOTIVAÇÃO

Embora várias VQL tenham sido propostas, uma revisão do estado da arte (cf. Capítulo 3) verificou que essas VQL não são amplamente utilizadas na prática, pois não abrangem várias construções complexas e não possuem ferramentas CASE disponíveis para seus usuários finais, comprometendo sua expressividade e disponibilidade. Isso é confirmado por Abouzied, Hellerstein e Silberschatz (2012) e Bakke e Karger (2016) que afirmam que, embora haja estudos recentes sobre VQL, pouco foi feito para melhorar desde as propostas iniciais. Além disso, destaca-se que na revisão não se encontrou uma VQL que:

- suporte ao menos 50% dos construtores complexos analisados;

- permita especificar expressões condicionais;
- permita especificar subconsultas nas cláusulas *Select*, *From*, *Having* e *Order By*.

Outro ponto que motiva este trabalho são as pesquisas recentes em diversas áreas que usaram amplamente o paradigma MDD para resolver diversos desafios. Por exemplo, a revista *Journal of Systems and Software* lançou uma edição focada no uso de MDD como solução para simplificar a engenharia reversa de software (FONTANA et al., 2020) e a revista *Computer Languages, Systems & Structures* lançou uma edição especifica para discutir sobre MDD aplicado a sistemas multi agentes (KARDAS; GOMEZ-SANZ, 2017). De forma semelhante, a área de Banco de Dados também pode usufruir deste paradigma para resolver os seus desafios. No entanto, no contexto de VQL, se desconhece uma solução que tenham sido desenvolvida com base no paradigma MDD.

1.3 OBJETIVO

O objetivo geral dessa tese é especificar uma VQL denominada *Diagrammatic Structured Query Language* (DSQL) que considere todas as construções complexas mencionadas e seja tão compreensível e eficaz quanto SQL sem aumento do esforço. Para alcançar esse objetivo geral têm-se quatro objetivos específicos. Os três primeiros são objetivos acadêmicos e o quarto é um objetivo tecnológico.

O primeiro objetivo acadêmico é realizar uma revisão sistemática do estado da arte de modo a investiga as limitações das VQL usadas para especificar consultas SQL complexas. Nessa revisão são respondidas as seguintes perguntas de pesquisa:

- PP1: qual é a abstração visual usada por cada VQL?
- PP2: quais são os construtores complexos suportados por cada VQL?
- PP3: qual é o suporte para especificar subconsultas ou expressões condicionais fornecidas por cada VQL?
- PP4: qual é o suporte de ferramenta CASE fornecido por cada VQL?
- PP5: qual é o método usado para avaliar cada VQL?

O segundo objetivo acadêmico é propor a notação (i.e., sintaxe concreta) da linguagem de modelagem de modo a cobrir as limitações encontradas na revisão.

O terceiro objetivo acadêmico é propor um metamodelo (i.e., sintaxe abstrata) para dar suporte a todos os construtores da notação proposta.

O objetivo tecnológico é a implementação da ferramenta CASE com base no metamodelo que dá suporte à notação proposta. Dentro do contexto da ferramenta CASE, como prova de conceito será especificada uma primeira versão das regras de boa formação (i.e., semântica estática) e das regras de transformação.

1.4 DELIMITAÇÃO DO ESCOPO

A definição sistemática e exaustivo da semântica estática e das regras para transformação dos diagramas DSQL em código SQL-DML (*Data Manipulation Language*) estão fora do escopo dessa tese. Contudo, uma ferramenta CASE que implementa uma proposta inicial destas regras foi desenvolvida. O propósito desta ferramenta é mostrar a viabilidade técnica e tecnológica deste trabalho. Portanto, podem existir regras de boa formação que não são identificadas e podem existir diagramas DSQL que não sejam transformadas no código SQL esperado e vice-versa.

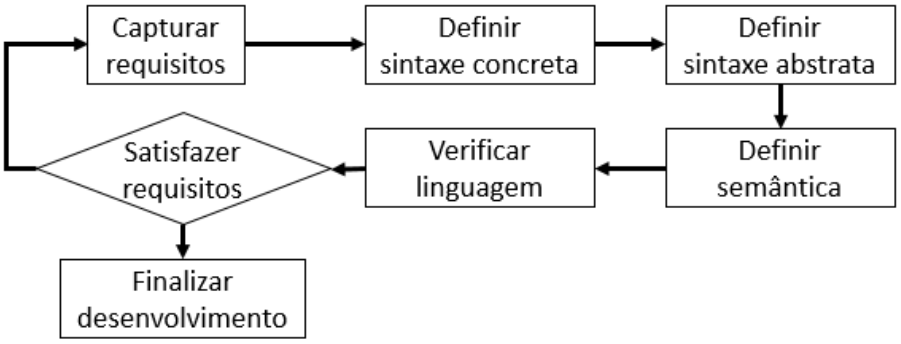
A avaliação da linguagem proposta limita-se a aspectos de expressividade e da compreensão de consultas complexas. Isto é, embora a avaliação dos aspectos de usabilidade (e.g., aprendizado, eficiência, memorização, satisfação (NIELSEN, 1995)) relacionadas a criação de consultas utilizando DSQL sejam importantes, estes estão além do escopo deste trabalho.

Embora ferramentas de visualização de dados (e.g., Tableau, Microsoft Power BI ou QlikView) permitam a especificação de consultas complexas, essas ferramentas não suportam esse tipo de construção sem a sobreposição de múltiplas interfaces, o que compromete a compreensão e a análise das consultas, pois os seus detalhes não estão visíveis em uma única visão da interface. Portanto, essas ferramentas não são consideradas como trabalhos relacionados.

1.5 METODOLOGIA

O desenvolvimento dessa tese inicia com uma revisão sistemática da literatura (cf. Capitulo 3), a qual visa entender porque as VQL não são amplamente utilizadas na prática. Em seguida, o paradigma MDD (cf. Seção 2.2) é usado como arcabouço teórico e tecnológico para a especificação de DSQL, a qual segue o processo de desenvolvimento proposto por Cho, Gray e Syriani (2012), apresentado na Figura 2. Nesse processo o desenvolvimento de uma *Domain-Specific Modeling Language* (DSML) começa a partir da captura dos requisitos. Nessa etapa, os especialistas do domínio descrevem quais competências serão requeridas e como isso será representado. Em seguida, com base nos requisitos, os especialistas em desenvolvimento de linguagem identificam a sintaxe concreta (i.e., a notação), projetam a sintaxe abstrata (i.e., os conceitos, atributos e relacionamentos entre os conceitos) e especificam a semântica (i.e., o significado dos conceitos e os comportamentos permitidos pela sintaxe abstrata que devem ser evitados). Ao fim dessas etapas, os especialistas do domínio verificam a linguagem de modelagem e provêm *feedback* informativo. Por fim, para avaliar a expressividade de DSQL é realizada uma análise comparativa com as propostas relacionadas e um experimento para comparar a precisão, o tempo e o esforço para compreender consultas complexas usando DSQL e SQL.

Figura 2 – Processo de desenvolvimento de uma DSML



Fonte: adaptado de Cho, Gray e Syriani (2012)

1.6 ESTRUTURA DO DOCUMENTO

O restante desse documento é dividido em 5 capítulos. No Capítulo 2 apresentam-se os fundamentos teóricos necessários para embasar esse trabalho, os quais são divididos em 3 seções: consulta SQL complexa, desenvolvimento dirigido por modelos e, por fim, linguagens visuais de consulta. No Capítulo 3 apresenta-se a revisão sistemática para entender o nível de suporte teórico e tecnológico que as VQL oferecem para especificar consultas SQL complexas. No Capítulo 4 apresenta-se a VQL proposta, ou seja, neste capítulo especifica-se a sintaxe concreta, a sintaxe abstrata e a ferramenta CASE que dá suporte a DSQL. No Capítulo 5 avalia-se a expressividade de DSQL em comparação as propostas relacionadas e faz-se dois experimentos para comparar a compreensão de consultas em DSQL e em SQL. O primeiro em relação ao tempo e precisão e o segundo em relação ao tempo, precisão e esforço. Por fim, no Capítulo 6 apresentam-se as considerações finais, contribuições, limitações e trabalhos futuros relacionados a este trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo apresenta-se a fundamentação teórica que serve de base para o entendimento deste trabalho. São discorridos conceitos sobre consulta SQL complexa, desenvolvimento dirigido por modelos e, por fim, linguagem visual de consulta.

2.1 CONSULTA SQL COMPLEXA

Existem diferentes percepções sobre quais construções tornam uma consulta SQL complexa (CATARCI; SANTUCCI, 1995; ZHANG et al., 1999; KAWASH, 2004; GRYZ et al., 2008; EL-MAHGARY; SOISALON-SOININEN, 2015). Nesse estudo, uma consulta é considerada complexa sempre que contém pelo menos uma das seguintes construções: subconsulta, junção, operação de conjunto, expressão condicional, restrição de agrupamento ou recursão. Esses construtores têm potencial para aumentar a complexidade de uma consulta, principalmente se isso envolve mais de um desses construtores. Nos parágrafos a seguir desta seção, uma visão geral dos aspectos mais importantes desses construtores (ELMASRI; NAVATHE, 2016) e exemplos de consultas SQL complexas (cf. Figura 6) são apresentados.

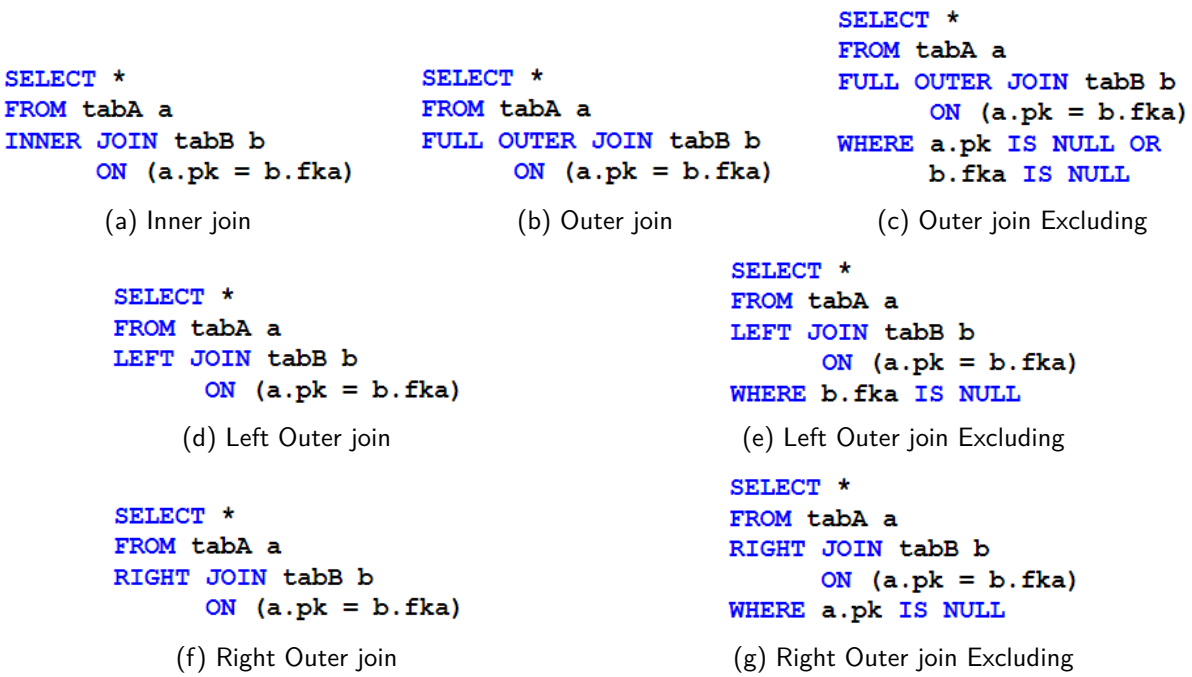
Existem dois tipos de subconsultas: (1) correlacionada, em que a subconsulta é avaliada uma vez para cada linha processada pela consulta externa, pois a subconsulta faz referência a pelo menos uma coluna de sua consulta externa; e (2) não correlacionado, na qual a subconsulta é executada apenas uma vez (antes de executar a consulta externa). As subconsultas podem ser especificadas nas seguintes cláusulas: *Select*, *From*, *Where*, *Having* e *Order By*.

Existem quatro tipos de junções: (1) junção cruzada (*Cross Join*), que retorna o produto cartesiano das linhas da tabela; (2) junção interna (*Inner Join*), que retorna apenas registros que satisfazem a condição de junção; (3) junção natural (*Natural Join*), um tipo de junção interna cuja condição de junção é feita implicitamente a partir da igualdade de campos com os mesmos nomes; e (4) junção externa (*Outer Join*), que estende os resultados de uma junção interna adicionando linhas da tabela esquerda (*Left Outer Join*), tabela da direita (*Right Outer Join*) ou ambas as tabelas (*Full Outer Join*) que não correspondem à condição de junção. Junções cruzadas, junções naturais e junções externas devem ser especificadas na cláusula *From*, enquanto junções internas podem ser especificadas nas cláusulas *From* ou *Where*. Observe que uma junção cruzada é uma junção que combina todas as linhas de duas tabelas; uma junção natural é uma junção interna usando uma condição de igualdade (*Equi Join*); e as junções internas e externas podem ser um *Equi Join*, um *Self Join* (uma junção que une uma tabela a ela mesma) ou um *Theta Join* (uma junção baseada em qualquer operador de comparação que não seja igual). Portanto, os principais tipos de junção são *Cross*, *Inner* e *Outer*.

De acordo com Moffatt (2009), pode-se obter outros tipos de *Outer Join* com o uso de

restrições de seleção, como: (1) (*Left Outer Join Excluding*), que retorna apenas as linhas da tabela esquerda que não satisfazem a condição de junção; (2) (*Right Outer Join Excluding*), que retorna apenas as linhas da tabela direita que não satisfazem a condição de junção; e (3) (*Full Outer Join Excluding*), que retorna todas as linhas que não satisfazem a condição de junção. Na Figura 3 apresentam-se exemplos genéricos de consultas SQL mostrando os tipos de junção.

Figura 3 – Tipos de junções SQL



(a) Inner join (b) Outer join (c) Outer join Excluding

(d) Left Outer join (e) Left Outer join Excluding

(f) Right Outer join (g) Right Outer join Excluding

Fonte: adaptado de Moffatt (2009)

Existem três operadores na álgebra relacional que são semelhante a junção. Esses operadores não têm representação exata em SQL, mas são especificados com auxilio de junção ou subconsulta (i.e., usando o operador *In* ou *Exists*). Esses operadores são: (1) Semi-junção (*Semi-join*) - retorna apenas as linhas de uma relação A que estão relacionadas com as linhas de uma relação B. Isto é, nenhum dado da tabela B é projetado. Esse operador pode ser especificado a partir de uma subconsulta correlacionada usando o operador *Exists* de SQL e, neste caso, tem performance melhor do que o *Inner Join*, pois, com o *Exists*, uma única condição verdadeira é suficiente, o que elimina a necessidade do modificador *Distinct*; (2) Anti-junção (*Anti-Join*) - retorna apenas as linhas em uma relação A que não possuem relacionamento com relação B. Esse operador pode ser especificado a partir da negação de uma operação de semi-junção e, por isso, tem performance superior *Left Outer Join Excluding*; e (3) Divisão (*Division*) - que retorna apenas as linhas de uma relação A que possui correspondência com todas as instâncias de uma relação B. Ressalta-se que para realizar uma divisão é necessário que os atributos de B sejam um subconjunto dos atributos de A (CELKO, 2010). A Figura 4 mostra exemplos de consultas SQL para estas operações da álgebra relacional. Na Figura 4a

apresenta-se uma consulta com semi-junção. Na Figura 4b apresenta-se uma consulta com anti-junção. Por fim, na figura 4c apresenta-se uma consulta com divisão.

Figura 4 – Exemplo de semi-junção, anti-junção e divisão em SQL

```
SELECT p.name
FROM product p
WHERE EXISTS
   (SELECT *
    FROM orderItem oi
    WHERE oi.productId = p.id)
```

(a) Quais produtos já foram vendidos?

```
SELECT p.name
FROM product p
WHERE NOT EXISTS
   (SELECT *
    FROM orderItem oi
    WHERE oi.productId = p.id)
```

(b) Quais produtos nunca foram vendidos?

```
SELECT DISTINCT pilot_name
FROM PilotSkills PS1
WHERE NOT EXISTS
     (SELECT *
      FROM Hangar
      WHERE NOT EXISTS
           (SELECT *
            FROM PilotSkills AS PS2
            WHERE (PS1.pilot_name = PS2.pilot_name) AND
                  (PS2.plane_name = Hangar.plane_name)))
```

(c) Quais pilotos podem pilotar todos os aviões?

Existem três tipos de operações de conjunto: (1) união (*Union*), que combina resultados de duas consultas, retornando por padrão apenas linhas distintas; (2) interseção (*Intersect*), que mescla os resultados de duas consultas, retornando por padrão apenas as linhas distintas comuns a ambas; e (3) exceto (*Except*), que corresponde aos resultados de duas consultas, retornando por padrão apenas as linhas da primeira consulta que não aparecem na segunda consulta. Deve-se enfatizar que o modificador *ALL* pode ser usado para retornar todas as linhas, em vez de apenas linhas distintas. Ressalta-se que além destes três tipos de operações de conjunto, de acordo com Date (2015), pode-se usar uma união exclusiva (*Exclusive Union*), que combina os resultados de duas consultas, retornando as linhas que aparecem somente em uma das consultas. Essa operação não tem um operador específico em SQL, mas pode ser especificada nessa linguagem. Na Figura 5 apresenta-se um exemplo de consulta SQL com união exclusiva. Nessa consulta pergunta-se: *"qual o nome das pessoas que apenas são médico ou paciente?"*.

Figura 5 – Exemplo de união exclusiva em SQL

```
    SELECT nome
    FROM medico
    WHERE nome NOT IN
       (SELECT nome
        FROM paciente)
UNION
    SELECT nome
    FROM paciente
    WHERE nome NOT IN
       (SELECT nome
        FROM medico)
```

Expressões condicionais (i.e., cláusulas *Case-When*) permitem que variáveis sejam testadas. Essas declarações de tomada de decisão podem ser especificadas nas seguintes cláusulas: *Select*, *Where*, *Order By* e *Having*.

As expressões de agrupamento (i.e., cláusula *Group By*) agrupam linhas com os mesmos valores em uma única linha. Se uma restrição de agrupamento precisa ser definida, esta deve ser especificada na cláusula *Having*.

No escopo da seção, *Common Table Expression* (CTE) (i.e., cláusula *With*) permitem a definição de uma visão (*View*) que pode ser recursivamente referenciada como uma fonte de dados quando adicionado o modificador *RECURSIVE*.

Figura 6 – Exemplos de consultas SQL complexas

```
SELECT p1.productId,p1.price -
    (SELECT AVG(p2.price)
     FROM product p2
     WHERE p2.model = p1.model)
FROM product p1
WHERE p1.price >
    (SELECT AVG(p2.price)
     FROM product p2
     WHERE p2.model = p1.model)
```

(a) Qual é o preço em excesso dos produtos cujos preços estão acima do preço médio dos produtos do mesmo modelo?

```
SELECT p1.model
FROM product p1
GROUP BY p1.model
HAVING AVG(p1.price) >
    (SELECT AVG(p2.price)
     FROM product p2)
```

(b) Quais são os modelos dos produtos com preço médio superior ao preço médio geral?

```
SELECT o.customerId,
    AVG(qi.qty) as meanQty
FROM order o
    INNER JOIN (
        SELECT orderId,
            COUNT(*) as qty
        FROM orderItem
        GROUP BY orderId) as qi
    ON (o.id = qi.orderId)
GROUP BY o.customerId
ORDER BY meanQty
```

(c) Para cada cliente, qual é o número médio de itens por pedido?

```
SELECT o.orderId,
   (CASE
         WHEN o.couponType = 'A' THEN o.value * 0.8
         WHEN o.couponType = 'B' THEN o.value * 0.9
         ELSE o.value
    END) - (SELECT SUM(o2.value) * 0.01
            FROM order o2
            WHERE o.customerId = o2.customerId) as discountedValue
FROM order o;
```

(d) Quais seriam os valores dos pedidos se os cupons A e B tivessem um desconto de 20% e 10%, respectivamente, e se um desconto adicional de 1% na soma dos pedidos dos clientes também fosse concedido?

```
WITH RECURSIVE
  cathdeph(name, supercat, level) AS
  (   SELECT name, supercat, 1 AS level
      FROM category
    UNION
      SELECT r.name, c.supercat, r.level + 1
      FROM cathdeph r
      INNER JOIN category c ON (r.supercat = c.id)
  )

SELECT name, MAX(level)
FROM cathdeph r
GROUP BY name
ORDER BY 2;
```

(e) Qual o maior nível de super categoria para cada categoria?

```
SELECT *
FROM product p
ORDER BY
   (SELECT COUNT(p.model)
    FROM product p2
    WHERE p2.model = p.model)
DESC
```

(f) Quais são os produtos ordenados pela quantidade de produtos do mesmo modelo?

A Figura 6 mostra seis exemplos de consultas SQL complexas. Na Figura 6a, uma subconsulta correlacionada é mostrada nas cláusulas *Select* e *Where*. Na Figura 6b, são apresentadas uma restrição de agrupamento e uma subconsulta não correlacionada (especificada na

cláusula *Having*). Na Figura 6c, uma junção interna e uma subconsulta não correlacionada (especificadas na cláusula *From*) são exibidas. Na Figura 6d, uma expressão condicional e uma subconsulta correlacionada (especificada na cláusula *Select*) são mostradas. Na Figura 6e, uma recursão e uma operação de conjunto são usadas. Por fim, na Figura 6f, subconsulta correlacionada é mostrada nas cláusulas *Order By*.

2.2 DESENVOLVIMENTO DIRIGIDO POR MODELOS

O desenvolvimento dirigido por modelos (*Model-Driven Development* - MDD) (BRAMBILLA; CABOT; WIMMER, 2017; KELLY; TOLVANEN, 2008) baseia-se no princípio de que modelos[1] são artefatos executáveis que contêm detalhes suficientes para a geração automática ou semi-automática de código. Isto é, modelos não são apenas artefatos de documentação, uma vez que estes podem ser automaticamente transformados em outros modelos (i.e., transformação modelo-modelo ou M2M - e.g., transformar um modelo de acordo com a notação do diagrama de classes da UML em um modelo de acordo com a notação do diagrama Entidade e Relacionamento de Peter Chen) e estes modelos podem ser transformados em código executável/interpretável (i.e., transformação modelo-texto ou M2T - e.g., transformar um diagrama de classe de UML em um código SQL-DDL para criação do esquema do banco de dados relacional).

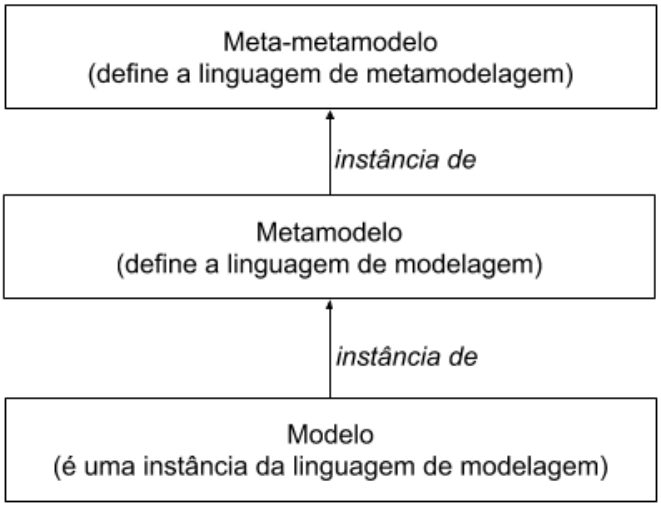
Uma maneira de usufruir do paradigma MDD é utilizar uma linguagem de modelagem específica de domínio (*Domain-Specific Modeling Language* - DSML) (BRAMBILLA; CABOT; WIMMER, 2017), pois essa, é dedicada ao domínio do problema do usuário, uma vez que usa notações e abstrações específicas que permitem aumentar a compreensão e a produtividade, bem como evitar ou alertar erros em um dado domínio.

A proposta de uma DSML consiste na especificação da sua sintaxe abstrata (i.e., metamodelo ou gramática), sintaxe concreta (notação gráfica ou representação visual) e semântica (significado, incluído regras de boa formação) (ACHILLEOS; GEORGALAS; YANG, 2007; KELLY; TOLVANEN, 2008; BRAMBILLA; CABOT; WIMMER, 2017). Um metamodelo descreve a sintaxe abstrata (regras básicas de boa formação sintática) de uma linguagem de modelagem e é definido em uma linguagem de metamodelagem. Isto é, um metamodelo é um modelo que especifica todos os conceitos de um domínio e como estes devem estar relacionados. Portanto, é um metamodelo que deve dar suporte à especificação de qualquer modelo (sintaticamente válido) de uma linguagem de modelagem. Em um nível mais abstrato, mas seguindo o mesmo raciocínio, tem-se o conceito de meta-metamodelo. Como mostrado na Figura 7, o modelo, o metamodelo e o meta-metamodelo formam um padrão de metamodelagem de três níveis. O padrão MDD pode ser aplicado várias vezes, construindo, assim, uma hierarquia de camadas de abstração, na qual os modelos na camada *n* são especificados usando a linguagem definida como metamodelo na camada *n+1* (ACHILLEOS; GEORGALAS; YANG, 2007; KELLY; TOLVANEN,

[1] Neste trabalho modelo, esquema e diagrama são considerados sinônimos.

2008; BRAMBILLA; CABOT; WIMMER, 2017).

Figura 7 – Metamodelagem em MDD



Fonte: adaptado de Cetinkaya e Verbraeck (2011)

Existem várias linguagens de metamodelagem, sendo as mais comuns a *Essential Meta Object Facility* (EMOF)(OMG, 2006) e a Eclipse ECore[2]. O metamodelo EMOF foi criado com base na especificação *Meta Object Facility* (MOF)[3] e seu principal objetivo é permitir a definição de metamodelos usando conceitos mais simples do que o MOF e o suporte a extensões. O meta-metamodelo Eclipse Ecore é a parte principal do projeto *Eclipse Modeling Framework* (EMF) (EMF, 2011). O projeto EMF é um *framework* de modelagem e gerador de código para a construção de ferramentas e outras aplicações com base em um modelo de dados estruturado. O Eclipse Ecore é baseado na especificação EMOF, com poucas diferenças, como, por exemplo, o uso do prefixo "E" na definição das construções de metamodelagem (i.e., *EClass*, *EAttribute*, *EReference* e *EDataType*).

Uma ferramenta de modelagem fornece formas de desenvolver modelos baseados em uma linguagem de modelagem. Na maioria dos casos, as ferramentas de modelagem são estendidas com ferramentas extras, como um transformador de modelo, um compilador, um tradutor, um verificador de consistência ou um depurador (na qual uma ferramenta é um instrumento para executar alguma tarefa). Nesse caso, o conjunto de ferramentas completo é chamado de ambiente de modelagem. A ferramenta resultante pode funcionar no ambiente de meta-modelagem ou, menos comumente, ser produzida como um programa autônomo separado. Os ambientes de metamodelagem mais comuns são as ferramentas: Microsoft DSL Tools[4], MetaEdit+[5] e *Graphical Modeling Project* (GMP) (BUDINSKY, 2004). Vale destacar que no

[2] https://wiki.eclipse.org/Ecore
[3] http://www.omg.org/spec/MOF/2.5.1/
[4] https://docs.microsoft.com/en-us/visualstudio/modeling/overview-of-domain-specific-language-tools
[5] https://www.metacase.com

ambiente GMP destacam-se os *frameworks* Epsilon[6] e Sirius[7], pois estes *frameworks* são uma família de linguagens e ferramentas que auxiliam na especificação do metamodelo, nas regras de validação, nas regras de transformações e na geração da ferramenta CASE.

É importante ressaltar que podem existir combinações sintaticamente válidas no metamodelo que, por razões de semântica, devem ser impedidas. Isto é, para que seja possível estabelecer um significado para determinadas construções sintaticamente corretas, é necessário definir restrições/condições relativas ao uso dos seus elementos sintáticos (KELLY; TOLVANEN, 2008). Essas regras de boa formação podem ser especificadas em *Object Constraint Language* (OCL) (OMG, 2014) ou em uma linguagem semelhante (e.g., *Epsilon Validation Language* (EVL)(KOLOVOS et al., 2017)).

2.3 LINGUAGEM VISUAL DE CONSULTA

As linguagens visuais de consulta (*Visual Query Language* – VQL) (CATARCI et al., 1997) podem ser definidas como linguagens para bancos de dados que usam representações visuais para representar o domínio de interesse e especificar solicitações para esse domínio. Portanto, as VQL são uma alternativa à SQL. Diferentemente das representações textuais, as abstrações visuais de uma VQL permitem que os usuários se concentrem nos aspectos semânticos ao invés dos aspectos sintáticos das consultas (CATARCI; SANTUCCI, 1995; HVORECKý; DRLíK; MUNK, 2010). Isso favorece o desenvolvimento do pensamento crítico e criativo, uma vez que as abstrações visuais das VQL criam condições que melhoram o pensamento analítico e as habilidades de resolução de problemas. Além disso, as abstrações das VQL permitem que as informações sejam interpretadas mais rapidamente do que a notação textual de SQL, pois podem explorar a capacidade do sistema visual humano de processar informações em paralelo (MOODY, 2009; WARE, 2004).

Embora seja possível usar uma VQL apenas com caneta e papel, uma VQL é mais útil quando suportada por uma ferramenta CASE (*Computer-Aided Software Engineering*), pois esse tipo de ferramenta facilita a interação com a notação da VQL e possui funcionalidades que podem evitar ou detectar erros, além de gerar artefatos de software (e.g., código fonte ou documentação) (BRAMBILLA; CABOT; WIMMER, 2017). Ressalta-se que uma ferramenta CASE pode ser desenvolvida seguindo um processo tradicional de desenvolvimento ou com base na abordagem MDD.

As abstrações visuais de uma VQL podem ser baseadas em quatro notações (CATARCI; SANTUCCI, 1995): formulário, diagrama, ícone ou híbrida (i.e., uma combinação arbitrária das notações acima). Uma notação baseada em formulário é uma generalização de uma tabela e pode representar relacionamentos entre células, subconjunto ou conjunto. Um exemplo de VQL baseada em formulário é *Query-By-Example* (QBE) (ZLOOF et al., 1975) (cf. Figura 8a). Uma notação baseada em diagrama usa componentes visuais (e.g., formas geométricas, cores,

[6] https://www.eclipse.org/epsilon
[7] https://www.eclipse.org/sirius

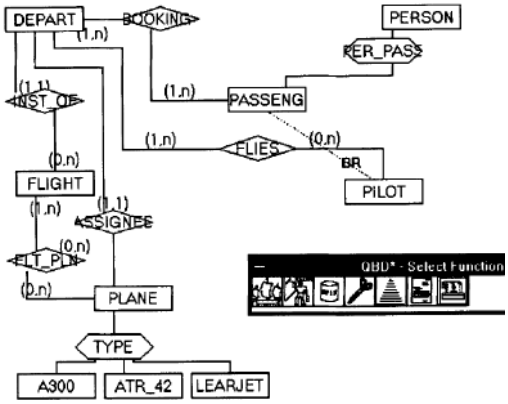
Figura 8 – Exemplos de VQL

EMP	NAME	SAL	MGR	DEPT
	P.JONES	P.10K	P.SMITH	TOY

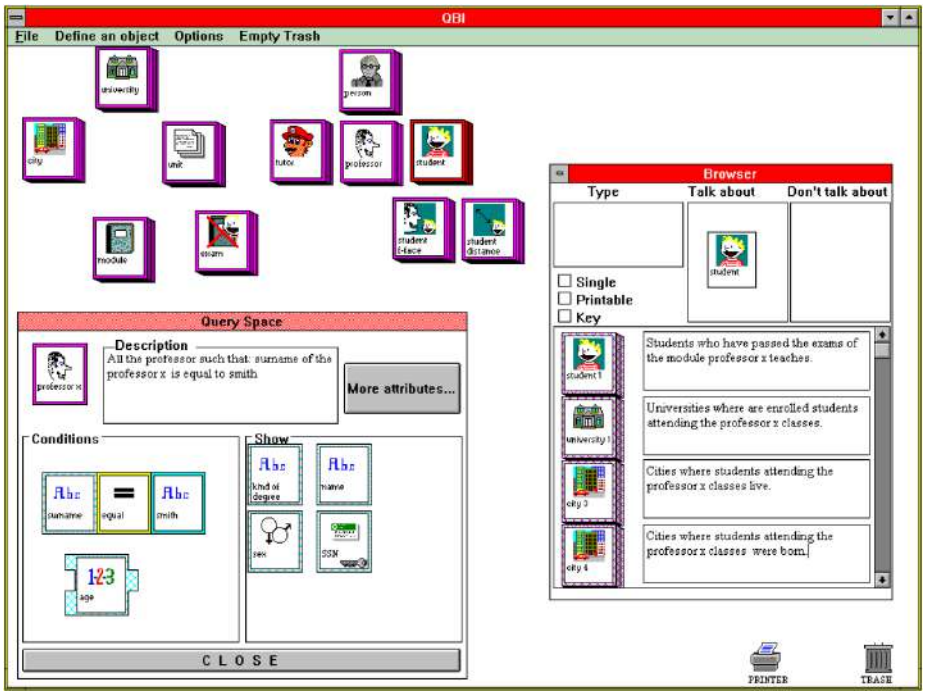
ANS:

NAME	SAL	MGR
ANDERSON	6K	MURPHY
NELSON	6K	MURPHY
HENRY	9K	SMITH

(a) Liste os nomes, salários e gerentes dos funcionários do departamento de brinquedos (ZLOOF et al., 1975)



(b) Encontre todos os pilotos que possam ser parentes dos passageiros reservados no voo X? (SANTUCCI; SOTTILE, 1993)



(c) Quem são os professores com sobrenome igual a Smith? (BADRE et al., 1996)

setas) para representar conceitos e relacionamentos entre eles. Um exemplo de VQL baseada em diagrama é *Query-by-Diagram* (QBD) (SANTUCCI; SOTTILE, 1993) (cf. Figura 8b). Uma notação baseada em ícone mapeia um conceito real em um ícone que representa uma abstração de uma categoria ou ideia. Um exemplo de VQL baseada em ícone é *Query-by-Icon* (QBI) (BADRE et al., 1996) (cf. Figura 8c).

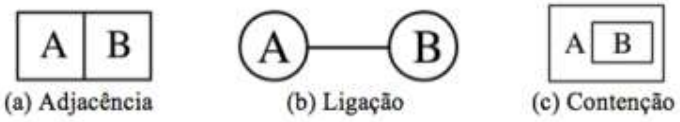
De acordo com Nickerson (1994), no contexto de VQL, os diagramas que retratam as associações dos elementos podem ser divididos em três categorias: métricas, simbólicas e topológicas. As associações métricas estão relacionadas com distâncias medidas entre objetos. Neste caso, pode ser utilizada qualquer escala para mensurar a distância. As associações simbólicas possuem um modo mais geral de serem representadas do que com o uso de descrições do

tipo: link(a,b), onde qualquer diagrama pode ser inteiramente descrito simbolicamente como uma lista textual de nós e arestas. E, por fim, a mais comumente utilizada é a classificação no espaço topológico (NICKERSON, 1994), na qual os elementos do diagrama se relacionam topologicamente entre si. Na categoria topológica, as associações podem acontecer nas seguintes formas:

- Adjacência - Quando A e B são células que compartilham um lado ou possuem lados sobrepostos;
- Ligação - Quando os nós A e B são explicitamente ligados por uma aresta. Grafos e árvores são formas de diagramas conhecidos que utilizam esta classificação;
- Contenção - Quando um nó está contido dentro do outro. Este método é mais utilizado para indicar relações estabelecidas, como em diagramas Venn;
- Híbrida - Combinação de duas ou mais associações mencionadas nos itens acima.

Na Figura 9 apresentam-se exemplos dos tipos de associações na classificação topológica de uma VQL. A representação chamada de Adjacência possui um limite de conectividade e, por isso, normalmente é utilizada em esquemas simples. Isto é, representar uma grande quantidade de associações e/ou complexidade alta com este tipo de ligação resulta em diagramas complexos e de difícil entendimento. Ao utilizar a representação Ligação, leva-se em conta o fato de que as arestas podem transmitir informações adicionais como, por exemplo, rótulos e direcionamento. Esta associação normalmente é utilizada em diagramas extensos e complexos devido à grande quantidade de informação que é possível transmitir com os relacionamentos entre os objetos. Por fim, a representação Contenção demonstra a relação estabelecidas entre os objetos no diagrama e, por isso, algumas estruturas podem ter um limite em sua representação. Por exemplo, estruturas recursivas são difíceis de serem representadas utilizando esta associação (NICKERSON, 1994).

Figura 9 – Classificação topológica para VQL



Fonte: adaptado de Nickerson (1994)

2.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Nesse capítulo, apresentaram-se os conceitos que formam a base para a compreensão dos demais capítulos dessa tese. Esse capítulo consiste de três temas: 1) consulta SQL complexa

- que defini quais são os conceitos SQL considerados complexos neste trabalho (i.e., sub-consulta, junção, operação de conjunto, expressão condicional, restrição de agrupamento e recursão); 2) desenvolvimento dirigido por modelos - que apresenta os conceitos de modelo, metamodelo, meta-metamodelo, transformação de modelo, metamodelagem, linguagem de modelagem específica de domínio e regras de boa formação; e 3) linguagem visual de consulta - que apresenta os pontos fortes do uso da abordagem e classifica as VQL quanto a notação visual. No próximo capítulo apresenta-se uma revisão sistemática da literatura que tem como propósito de investigar por que as VQL não são amplamente utilizadas na prática.

3 REVISÃO SISTEMÁTICA DA LITERATURA

Nesse estudo, uma Revisão Sistemática da Literatura (RSL) é realizada com base nas diretrizes definidas por Kitchenham e Charters (2007), que envolvem as seguintes etapas: 1) especificação das perguntas de pesquisa (cf. Seção 3.1); 2) desenvolvimento do protocolo de revisão (cf. Seções 3.2–3.5); 3) seleção dos artigos (cf. Seção 3.6); 4) avaliação da qualidade (cf. Tabela 1); e 5) análise e síntese de dados (cf. Seções 3.7–3.8).

3.1 PERGUNTAS DE PESQUISA

O objetivo dessa RSL é investigar a expressividade notacional, o suporte tecnológico e a avaliação científica das *Visual Query Language* (VQL) para investiga as limitações das VQL usadas para especificar consultas SQL complexas. Para atingir esse objetivo, a notação gráfica de cada VQL será analisada para responder às cinco perguntas de pesquisa (PP) a seguir:

- *PP1: qual é a abstração visual usada por cada VQL?*
  Responder essa pergunta de pesquisa é importante para saber qual notação gráfica (e.g., diagrama, formulário ou ícone) é mais comum.

- *PP2: quais são os construtores complexos suportados por cada VQL?*
  Nessa pergunta de pesquisa, a presença de cada construtor complexo (i.e., subconsulta, junção, operação de conjunto, expressão condicional, restrição de agrupamento e recursão) é apurada para determinar a expressividade das VQL e conhecer a frequência de cada construtor complexo.

- *PP3: qual é o suporte para especificar subconsultas ou expressões condicionais fornecidas por cada VQL?*
  Nessa pergunta de pesquisa, para cada cláusula SQL (i.e., *Select*, *From*, *Where*, *Order by* e *Having*), é analisado se o VQL fornece suporte para especificar uma subconsulta ou expressão condicional nela. Essa pergunta também é importante para determinar a expressividade de cada VQL, pois, diferentemente de outras construções complexas, subconsultas e expressões condicionais podem ser especificadas em mais de uma cláusula SQL.

- *PP4: qual é o suporte de ferramenta CASE fornecido por cada VQL?*
  Considerando que a disponibilidade de uma ferramenta CASE afeta diretamente o uso da VQL, nessa pergunta verifica-se o suporte de ferramenta CASE (i.e., disponível, indisponível ou ausente) para cada VQL.

- *PP5: qual é o método usado para avaliar cada VQL?*
  Responder essa pergunta de pesquisa é importante para saber qual método (e.g., pesquisa, entrevista ou experimentos controlados) é usado para avaliar cada VQL.

3.2 EXPRESSÃO DE PESQUISA E BIBLIOTECAS SELECIONADAS

A expressão de pesquisa usada para executar a RSL foi:

> "(SQL AND ("complex query" OR "complex queries" OR subquery OR subqueries) AND (visual OR graphic OR graphical))"

O termo *SQL* foi usado para restringir a pesquisa a essa linguagem. Os termos *complex query*, *complex queries*, *subquery* e *subqueries* direcionaram a pesquisa para propostas que abordam construções complexas. Os termos *visual*, *graphic* e *graphical* foram usados para limitar a pesquisa a abordagens visuais. A expressão de pesquisa considerou apenas o termo *subquery*, porque *join*, *having*, *case* e *with* são termos muito vagos, enquanto que *conditional expression* e *recursive* são muito restritivos. As bibliotecas pesquisadas foram: ScienceDirect, SpringerLink, ACM Digital Library e IEEE Xplore, pois essas bibliotecas incluem periódicos e conferências de alta qualidade em ciência da computação.

3.3 CRITÉRIOS DE INCLUSÃO E EXCLUSÃO

Essa revisão incluiu artigos publicados entre 1 de janeiro de 1980 e 28 de fevereiro de 2019, usando estes critérios de inclusão:

- *CI1* – artigos que apresentam notação gráfica;
- *CI2* – artigos que abordam a especificação de subconsulta.

Os seguintes critérios foram utilizados para excluir artigos:

- *CE1* – artigos secundários que revisam apenas uma área temática (e.g., uma RSL);
- *CE2* – artigos curtos (e.g., pôsteres, resumos ou *banners*);
- *CE3* – artigos redundantes do mesmo autor. Nesse caso, apenas o artigo mais completo e recente é mantido;
- *CE4* – artigos não escritos em inglês;
- *CE5* – literatura cinzenta, incluindo artigos não revisados por pares (e.g., relatórios, livros, teses ou dissertações).

Os critérios de inclusão e exclusão foram avaliados da seguinte forma: cada artigo foi revisado por três pesquisadores que leram o título, as palavras-chave e o resumo para determinar a relevância do artigo de acordo com cada critério. Quando necessário, o conteúdo do artigo também foi examinado. Em seguida, outros dois pesquisadores, independentemente, realizaram testes de sanidade. Finalmente, quaisquer diferenças que surgiram foram reconciliadas de maneira colaborativa.

3.4 CRITÉRIOS DE QUALIDADE

Para avaliação da qualidade, a relevância dos artigos foi verificada através da análise de dois critérios 1) o CORE *Conference/Journal Ranking*[1] e 2) a contagem de citações (Google Scholar). No CORE *Ranking*, uma conferência/periódico é atribuída a uma das seguintes categorias: A+, A, B, C ou *Unranked* (sem classificação). O artigo pontuou 4 se seu CORE *Ranking* foi classificado como A+, 3 se A, 2 se B, 1 se C e 0 se *Unranked*. Para as citações, o artigo obteve um 0, independentemente de não ter citações e 0,5 para cada dez citações. Por exemplo, um artigo recebeu uma pontuação de 0,5 se tinha entre 1 e 10 citações e recebeu uma pontuação de 1 se tinha entre 11 e 20 citações. O escore de qualidade de um artigo corresponde à soma dos escores obtidos na análise do CORE *Ranking* e das citações. Essa pontuação é importante porque ajuda a classificar os artigos.

3.5 EXTRAÇÃO DE DADOS

Após a seleção dos artigos, os dados foram extraídos dos estudos incluídos utilizando um formulário de extração. Esse formulário ajudou na captura de dados gerais (e.g., autor, ano, tipo de publicação) e na obtenção de dados específicos para cada questão de pesquisa. Nesse sentido, enquanto PP1, PP4 e PP5 são baseados em uma análise simples (apenas um critério), PP2 e PP3 são baseados nos critérios listados abaixo.

- Critérios (i.e., construções complexas) para responder à PP2:
  - *C1* – subconsulta não correlacionada;
  - *C2* – subconsulta correlacionada;
  - *C3* – junção cruzada;
  - *C4* – junção interna;
  - *C5* – junção externa;
  - *C6* – operação de conjunto;
  - *C7* – restrição de agrupamento;
  - *C8* – expressão condicional;

[1] http://www.core.edu.au/

  - *C9* – recursão.

- Critérios (i.e., cláusulas SQL) para responder à PP3:
  - *C10* – subconsulta na cláusula *Select*;
  - *C11* – subconsulta na cláusula *From*;
  - *C12* – subconsulta na cláusula *Where*;
  - *C13* – subconsulta na cláusula *Order By*;
  - *C14* – subconsulta na cláusula *Having*;
  - *C15* – expressão condicional na cláusula *Select*;
  - *C16* – expressão condicional na cláusula *Where*;
  - *C17* – expressão condicional na cláusula *Order By*;
  - *C18* – expressão condicional na cláusula *Having*.

Os critérios para PP2 e PP3 são binários e pontuam 0 ou 1 ponto. É importante destacar que os critérios para junção externa (C5) e operação de conjunto (C6) pontuam 1 ponto se a VQL suporta pelo menos um tipo de junção externa (*Outer*, *Left* ou *Right*) ou um tipo de operação de conjunto (*Union*, *Except* ou *Intersect*).

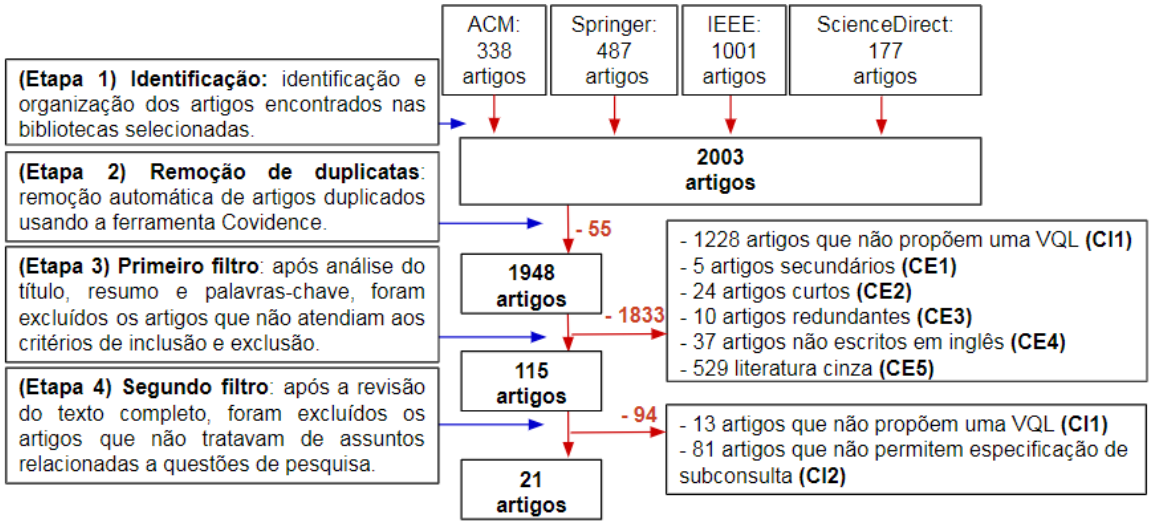
#	Autor	Título	Ano	CORE Rank	Citações	Escore
A01	Bloesch e Halpin	Conceptual queries using ConQuer-II	1997	A	131	10
A02	Tansel, Arkun e Ozsoyoglu	Time-by-example query language for historical databases	1989	A+	85	8.5
A03	Mohan e Kashyap	A Visual Query Language for Graphical Interaction with Schema-Intensive Databases	1993	A+	61	7.5
A04	Whang et al.	Two-dimensional specification of universal quantification in a graphical database query language	1992	A+	32	6
A05	Abouzied, Hellerstein e Silberschatz	DataPlay: Interactive Tweaking and Example-driven Correction of Graphical Database Queries	2012	A	43	5.5
A06	Murray et al.	Kaleidoquery A Flow-based Visual Language and its Evaluation	2000	A	41	5.5
A07	Klug	Abe: A Query Language for Constructing Aggregates-by-example	1981	A	42	5.5
A08	Danaparamita e Gatterbauer	QueryViz: Helping Users Understand SQL Queries and Their Patterns	2011	A	36	5
A09	Epstein	The TableTalk query language	1991	A	24	4.5
A10	Song, Yin e Ray	Using UML to model relational database operations	2007	B	38	4
A11	Jaakkola e Thalheim	Visual SQL – High-Quality ER-Based Query Treatment	2003	A	20	4
A12	Sockut et al.	GRAQULA: A graphical query language for entity-relationship or relational databases	1993	B	36	4
A13	Mun-Kew et al.	The implementation of a visual language interface for an object-oriented multimedia database system	1990	A	3	3.5
A14	Bakke e Karger	Expressive Query Construction Through Direct Manipulation of Nested Relational Results	2016	B	13	3
A15	Cerullo e Porta	A System for Database Visual Querying and Query Visualization: Complementing Text and Graphics to Increase Expressiveness	2007	B	3	2.5
A16	Tan, Chan e Siau	A graphical knowledge level approach for user-database interaction	1990	B	4	2.5
A17	Borges e Macías	Feasible Database Querying Using a Visual End-user Approach	2010	Unranked	14	1
A18	Keramopoulos, Pouyioutas e Sadler	GOQL, A Graphical Query Language for Object-Oriented Database Systems	1997	Unranked	18	1
A19	Keim e Lum	Visual Query Specification in a Multimedia Database System	1992	Unranked	16	1
A20	Ramos	Design and Implementation of a Graphical SQL with Generic Capabilities	1993	Unranked	4	0.5
A21	Kwak e Moon	Two-dimensional specification of queries in object-oriented databases	1995	Unranked	0	0

Tabela 1 – Avaliação da qualidade dos artigos selecionados

3.6 VISÃO GERAL DA EXECUÇÃO DA RSL

Esta seção apresenta uma visão geral da execução da RSL. Na Figura 10 é apresentado o processo de seleção dos artigos. Essa figura mostra as bibliotecas digitais, as etapas executadas, os artigos resultantes de cada etapa e os critérios utilizados para executar a RSL. Durante a identificação (Etapa 1), foram encontrados 2.003 artigos, que foram selecionados e organizados com a ferramenta Covidence[2]. Durante a remoção de duplicatas (Etapa 2), usando a mesma ferramenta, 55 artigos duplicados foram automaticamente identificados e excluídos, deixando um conjunto de 1.948 artigos. No primeiro filtro (Etapa 3), 1.833 artigos foram excluídos após a revisão de seus títulos, resumos e palavras-chave (cf. motivos na Etapa 3 da Figura 10), usando os critérios de inclusão e exclusão apresentados na Seção 3.3, deixando um conjunto de 115 artigos. Por fim, no segundo filtro (Etapa 4), após uma análise completa do texto, foram excluídos 94 artigos por não estarem de acordo com os critérios de inclusão (cf. motivos na Etapa 4 da Figura 10), deixando um total de 21 artigos (cf. Tabela 1).

Figura 10 – Processo para seleção dos artigos



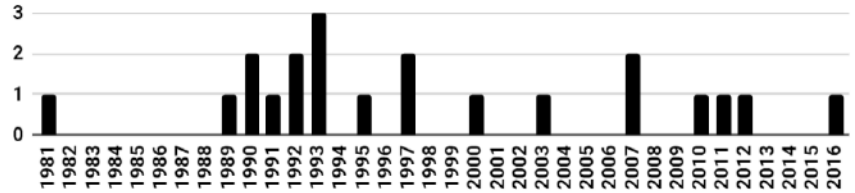
Na Tabela 1, é mostrada a avaliação da qualidade dos artigos selecionados. Nessa avaliação, o escore de qualidade médio de 4,04 indica que ainda há espaço para desenvolver estudos sobre a especificação de consultas SQL complexas usando VQL. Além disso, a maioria dos artigos é relevante porque foram publicados em boas conferências/periódicos (mais da metade dos artigos têm o CORE *Rank* A ou A+) e possuem média de 31,1 citações por artigo, com mediana de 24 citações.

Na Figura 11, é apresentada a frequência de publicações por ano. Os artigos foram publicados entre 1981 e 2016 e não mostram consistência ao longo do tempo em relação ao número de publicações que abordam consultas complexas. Pode-se observar que 1993 teve o

[2] https://www.covidence.org – uma ferramenta de metanálise para auxiliar na triagem dos resultados da pesquisa para identificar os estudos incluídos, realizar avaliação de risco de viés e extração de dados.

maior número de publicações (i.e., três artigos) e que nos últimos dez anos seis artigos foram publicados.

Figura 11 – Número de artigos publicados por ano



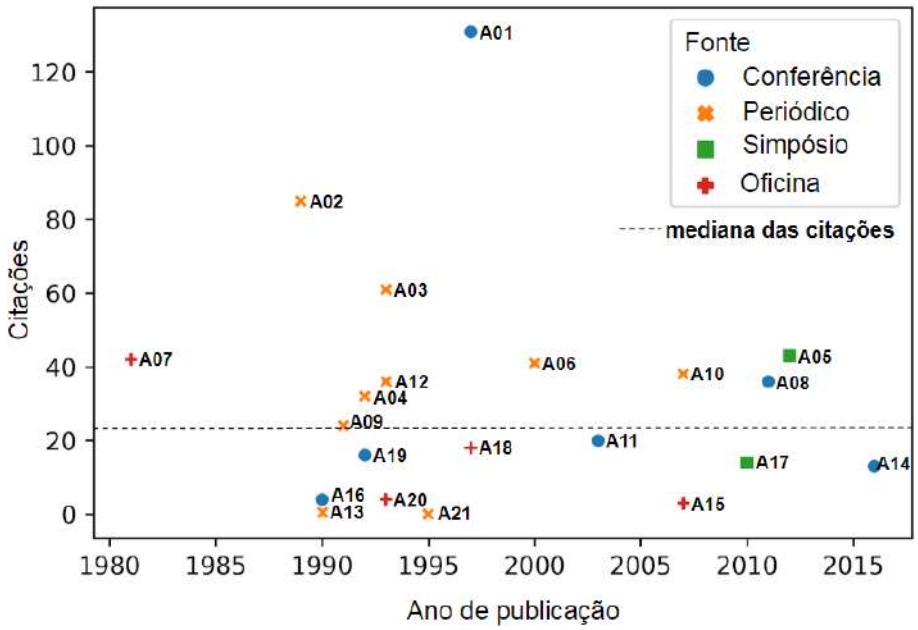
Fonte de publicação	Artigos	Quant.
Journal of Visual Languages and Computing	A06, A09, A13	3
IEEE Transactions on Software Engineering	A02, A04	2
International Conference on Conceptual Modeling	A01, A11	2
Computer Standards & Interfaces	A10	1
Conference on Visualization	A19	1
Data & Knowledge Engineering	A12	1
IEEE Transactions on Knowledge and Data Engineering	A03	1
International Computer Software and Applications Conference	A16	1
International Conference on Extending Database Technology	A08	1
International Conference on Management of Data	A14	1
International Workshop on Database and Expert Systems Applications	A15	1
International Workshop on Information Technology	A18	1
International Workshop on Interfaces to Database Systems	A20	1
Microprocessing and Microprogramming	A21	1
Symposium on Engineering Interactive Computing Systems	A17	1
Symposium on User Interface Software and Technology	A05	1
Workshop on Statistical Database Management	A07	1

Tabela 2 – Distribuição de artigos por fonte de publicação

Na Tabela 2, é mostrado o número de artigos selecionados por fonte de publicação. Nessa tabela os 21 artigos selecionados estão distribuídos em 17 fontes de publicação. Essa distribuição sugere que "a especificação de consultas SQL complexas usando VQL" é um tópico de interesse para vários veículos de publicação, especialmente o *Journal of Visual Languages and Computing*, com três artigos publicados. A Figura 12 mostra o gráfico de dispersão dos artigos quanto ao número de citações, ano de publicação e fonte de publicação. Como pode ser visto nessa figura, os artigos incluídos nessa revisão são provenientes de conferências, periódicos, oficinas e simpósios. A maioria dos artigos é publicada em periódicos (42,85%, 9 artigos), seguidos de conferências (28,57%, 6 artigos), oficinas (19,04%, 4 artigos) e simpósios (9,52%, 2 artigos). Além disso, o gráfico na Figura 12 mostra uma linha pontilhada que representa a mediana das citações (i.e., 24 citações). Observe que, dos artigos acima da mediana das citações, os artigos A05, A08 e A10 são mais proeminentes porque são os mais recentes e possuem uma quantidade considerável de citações. Por sua vez, considerando os artigos abaixo

da mediana das citações, os artigos A13, A16, A18, A19, A20 e A21 são menos proeminentes por serem os mais antigos e com poucas citações.

Figura 12 – Relação entre número de citações, ano de publicação e fonte de publicação



3.7 ANALISANDO OS ARTIGOS SELECIONADOS

Para mostrar evidências que ajudam a responder às questões de pesquisa desse trabalho, é apresentado a seguir um resumo estruturado destacando os pontos fortes de cada artigo. O resumo é guiado por dois exemplos de execução de consultas SQL (cf. Figuras 13a e 13b) e abrange os critérios de extração de dados mostrados na Seção 3.5. Os exemplos de execução são utilizados ao longo desse trabalho porque os exemplos apresentados nos artigos selecionados são muito específicos (i.e., não são adequados para realizar análises comparativas imparciais). Portanto, para fazer uma análise comparativa justa, os exemplos de execução são o mais básico possível e baseados no construtor de subconsulta, pois não é uma restrição basal (i.e., junção, operação de conjunto ou restrição de agrupamento) nem sofisticada (i.e., expressão condicional ou recursão). As figuras 13a e 13b mostram o uso de subconsultas não correlacionadas e correlacionadas, respectivamente. É importante destacar que duas ou mais tabelas na cláusula *From*, com ou sem uma condição de junção, são consideradas como construções de junção interna e junção cruzada, respectivamente. Além disso, para verificar como cada VQL especifica subconsultas correlacionadas, observa-se se a VQL: 1) permite a especificação de uma condição de seleção com alias na consulta interna; ou 2) possui um construtor específico para subconsultas correlacionadas, porque sem esses dois recursos básicos, dificilmente uma subconsulta correlacionada pode ser especificada. Por fim, para cada artigo analisado, são apresentadas figuras que reúnem e descrevem os exemplos relatados no artigo, destacando em azul cada evidência encontrada.

Figura 13 – Exemplos de execução de consultas SQL

```
SELECT name
FROM employee
WHERE salary > (SELECT AVG(salary)
                FROM employee)
```

(a) Quem são os funcionários com salário superior ao salário médio geral?

```
SELECT e.name
FROM employee e
WHERE e.salary > (SELECT AVG(e2.salary)
                  FROM employee e2
                  WHERE e.dep = e2.dep)
```

(b) Quem são os funcionários com salário superior ao salário médio de seu departamento?

Figura 14 – Exemplos de execução em ConQuer-II

```
✓Employee
    +- has Salary > AVG(Salary) for Employe
```

(a) Quem são os funcionários com salário superior ao salário médio geral?

```
✓Employee
    +- works Dep1
    +- has Salary > AVG(Salary) for Employe
                          +- works Dep2 = Dep1
```

(b) Quem são os funcionários com salário superior ao salário médio de seu departamento?

Figura 15 – Principais recursos de ConQuer-II

```
✓Academic
    +- is Professor
    |       +- holds Chair = Informatics
    +- not was awarded Degree in Year
                          +- is from University = UQ
C1 - Subconsulta não correlacionada

select X1.empnr
from Academic X1
where X1.chair = 'Informatics' and
    X1.empnr not in (select X2.empnr from Awarded X2
                     where X2.university = 'UQ')
```

(a) Quem são os acadêmicos que são professores de informática e não possuem um diploma da universidade UQ?

```
✓Employee
    +- owns Car1
    +- not drives Car1
                  +- count(Car1) for Employee > 1
C2 - Subconsulta correlacionada   C7 - Restrição de agrupamento

select X1.empnr
from Owns X1
where X1.empnr not in (
    select X2.empnr
    from Drives X2
    where X2.car in (select X3.car from Owns X3
                     where X3.empnr = X1.empnr)
    group by X2.empnr
    having count(X2.car) > 1)
```

(b) Quais funcionários têm mais de um carro e não dirigem nenhum desses carros?

```
✓Academic
    +- was awarded ✓Degree in Year
                          +- has Rating > 5
C4 - Junção interna

select X1.empnr, X1.degreecode, X1.university
from Awarded X1, Degree X2
where X1.degreecode = X2.degreecode and
    X1.university = X2.university and
    X2.rating > 5
```

(c) Quem são os acadêmicos, bem como seus diplomas, com uma classificação superior a 5?

```
✓Academic
    +- maybe was awarded ✓Degree in Year
                          +- has Rating > 5
C5 - Junção externa

select X1.empnr, X2.degreecode, X2.university
from Academic X1 left outer join (
        select * from Awarded X2, Degree X3
        where X2.degreecode = X3.degreecode and
            X2.university = X3.university and
            X3.rating > 5)
    on X1.empnr = X2.empnr
```

(d) Quem são os acadêmicos, bem como seus graus (se houver) com uma classificação maior que 5?

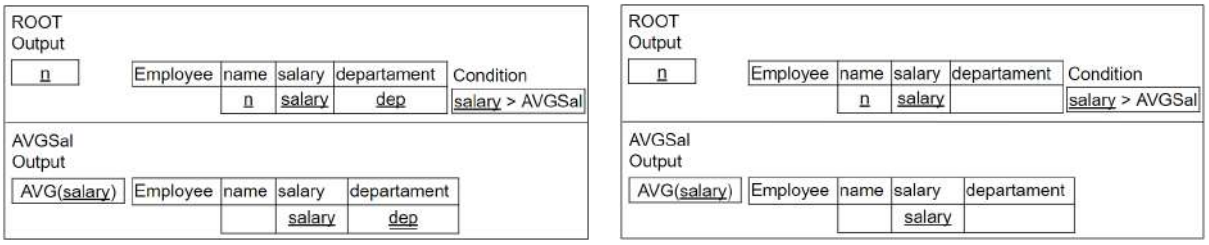
Fonte: adaptado de Bloesch e Halpin (1997)

ConQuer-II (A01)

A ConQuer-II (BLOESCH; HALPIN, 1997) permite a especificação de subconsultas não correlacionadas (C1) e correlacionadas (C2) (cf. Figuras 14a e 14b). Para isso, as condições de junção são implícitas e as subconsultas são especificadas ao definir uma condição de seleção entre duas entidades. Para especificação de subconsulta correlacionada, a ConQuer-II permite usar alias e condições de seleção na consulta interna. A ConQuer-II é baseado em *Object-Role Modeling* (ORM) (HALPIN; BLOESCH, 1999), uma abordagem de modelagem conceitual que mostra as aplicações em termos de objetos que desempenham papéis individualmente ou em relacionamentos. Portanto, os atributos *Employee works* e *has* significam acesso aos valores

dep e *salaray*, respectivamente. Em relação à PP1, a ConQuer-II é uma VQL baseada em notação de formulário e ícone. Em relação à PP2, as figuras 15a, 15b, 15c e 15d mostram o uso de subconsulta não correlacionada (C1), subconsulta correlacionada (C2), junção interna (C4), junção externa (C5) e restrição de agrupamento (C7). Em relação à PP3, as figuras 15a e 15b mostram uma subconsulta na cláusula *Where* (C12). Em relação à PP4, a linguagem é suportada por uma ferramenta CASE, mas não está disponível para download. Em relação à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo considera os critérios C1, C2, C4, C5, C7 e C12.

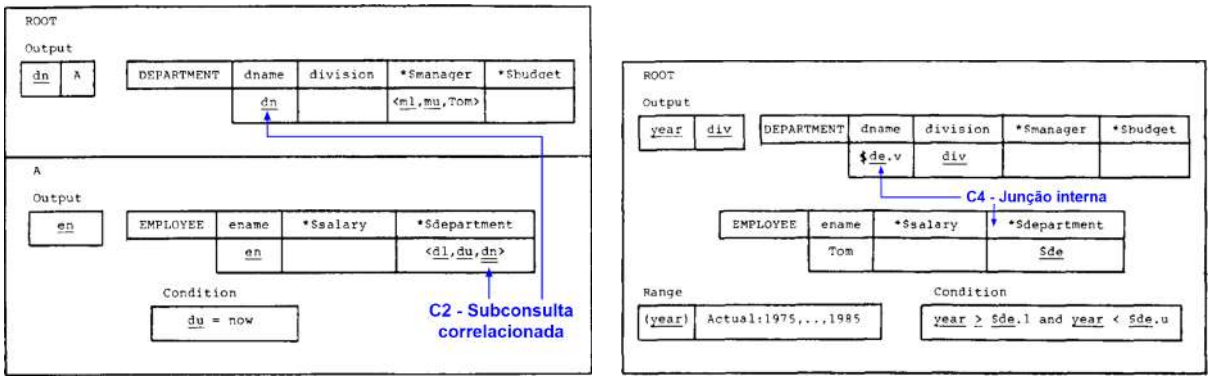
Figura 16 – Exemplos de execução em Time by Example (TBE)



(a) Quem são os funcionários com salário superior ao salário médio geral?

(b) Quem são os funcionários com salário superior ao salário médio de seu departamento?

Figura 17 – Principais recursos de Time by Example (TBE)



(a) Quais funcionários estão presentes em cada departamento gerenciado por Tom?

(b) Para quais divisões Tom trabalhou?

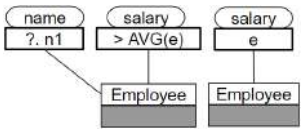
Fonte: adaptado de Tansel, Arkun e Ozsoyoglu (1989)

Time by Example (A02)

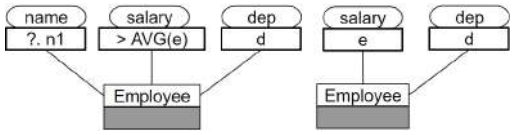
A Time by Example (TBE) (TANSEL; ARKUN; OZSOYOGLU, 1989) permite a especificação de subconsultas não correlacionadas (C1) e correlacionadas (C2) (cf. Figuras 16a e 16b). As subconsultas são definidas como blocos, com o bloco da consulta externa nomeado pela palavra reservada ROOT e o bloco de consulta interno definido abaixo do bloco ROOT (i.e., bloco *AVGSal*). Para especificação de subconsulta correlacionada, a TBE também usa alias (i.e., o atributo de *departament* como *dep*) e condições de seleção na consulta interna. Em

relação à PP1, a TBE é uma VQL baseada em notação de formulário. Em relação à PP2, as Figuras 17a e 17b mostram o uso de subconsulta correlacionada (C2) e junção interna (C4). Em relação à PP3, a Figura 17a mostra uma subconsulta na cláusula *Where* (C12). Em relação à PP4, a linguagem não possui ferramenta CASE. Em relação à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo considera os critérios C1, C2, C4 e C12.

Figura 18 – Exemplos de execução na linguagem proposta por Mohan e Kashyap



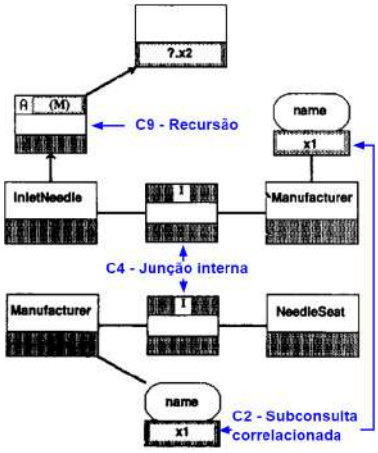
(a) Quem são os funcionários com salário superior ao salário médio geral?



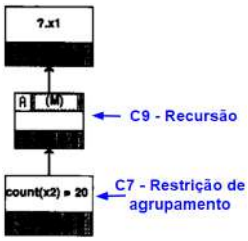
(b) Quem são os funcionários com salário superior ao salário médio de seu departamento?

Figura 19 – Principais recursos da linguagem proposta por Mohan e Kashyap

(a) Quais são os nomes dos itens que não são verdes?



(b) Quais são os componentes que usam (ou têm componentes que usam) *inlet needles* fabricadas por fabricantes que também fazem *needle seats*?



(c) Quais são os componentes que, juntamente com todos os componentes que os formam, têm mais de 20 componentes?

Fonte: adaptado de Mohan e Kashyap (1993)

Mohan and Kashyap (A03)

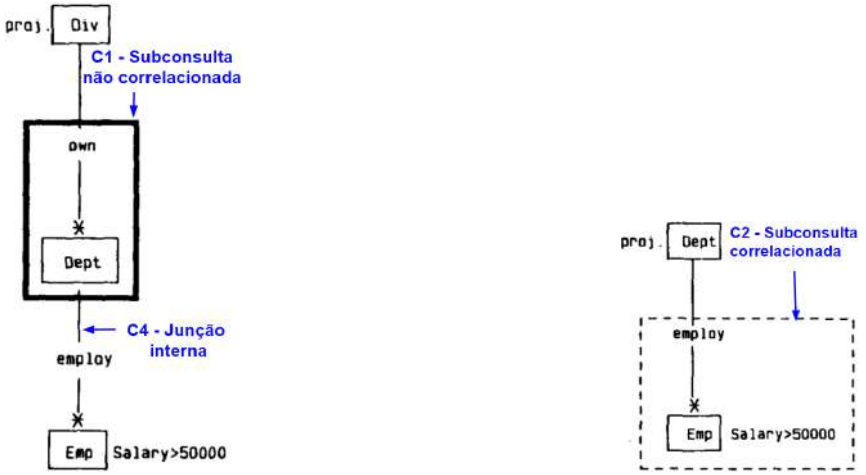
A linguagem proposta por Mohan e Kashyap (MOHAN; KASHYAP, 1993) permite a especificação de subconsultas não correlacionadas (C1) e correlacionadas (C2) (cf. Figuras 18a e 18b). Ambos os tipos de subconsultas podem ser definidos usando alias. Em uma subconsulta não correlacionada, um alias pode ser usado para especificar uma condição de seleção (i.e., o atributo *salary* como *e*). Da mesma forma, em uma subconsulta correlacionada, um alias pode ser usado para especificar a condição de seleção na consulta interna (i.e., o atributo *dep* como *d*). Em relação à PP1, essa linguagem é uma VQL baseada em notação de diagrama. Em relação à PP2, as figuras 19a, 19b e 19c mostram o uso de subconsulta não correlacionada

(C1), subconsulta correlacionada (C2), junção interna (C4), restrição de agrupamento (C7) e recursão (C9). Em relação à PP3, as figuras 19a e 19b mostram uma subconsulta na cláusula *Where* (C12). Em relação à PP4, a linguagem é suportada por uma ferramenta CASE, mas não está disponível para download. Em relação à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo considera os critérios C1, C2, C4, C7, C9 e C12.

Whang et al. (A04)

A linguagem proposta por Whang et al. (WHANG et al., 1992) permite a especificação de subconsultas não correlacionadas (C1) e correlacionadas (C2), mas os exemplos de execução não podem ser apresentados porque as funções de agregação estão além do escopo dessa linguagem (WHANG et al., 1992). Em relação à PP1, essa linguagem é uma VQL baseada em notação de diagrama. Em relação à PP2, as figuras 20a e 20b mostram o uso de subconsulta não correlacionada (C1), subconsulta correlacionada (C2) e junção interna (C4). Em relação à PP3, as figuras 20a e 20b mostram uma subconsulta na cláusula *Where* (C12). Em relação à PP4, a linguagem não possui ferramenta CASE. Em relação à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo considera os critérios C1, C2, C4 e C12.

Figura 20 – Principais recursos da linguagem proposta por Whang et al.



(a) Liste as divisões em que todos os seus departamentos possuem ao menos um funcionário cujo salário é superior a 50.000.

(b) Liste os departamentos em que nenhum funcionário do departamento tem salário superior a 50.000.

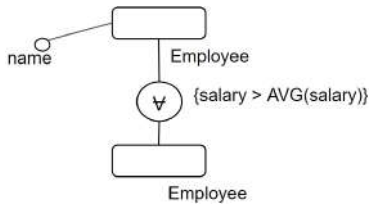
Fonte: adaptado de Whang et al. (1992)

DataPlay (A05)

A DataPlay (ABOUZIED; HELLERSTEIN; SILBERSCHATZ, 2012) permite a especificação de subconsultas não correlacionadas (C1) (Figura 21), mas não permite subconsultas correlacionadas porque não suporta nenhum dos dois recursos básicos para especificar uma correlação (i.e., uma condição de seleção com alias na consulta interna ou um construtor específico - cf. Seção 3.7). Em relação à PP1, a DataPlay é uma VQL baseada na notação de diagrama e

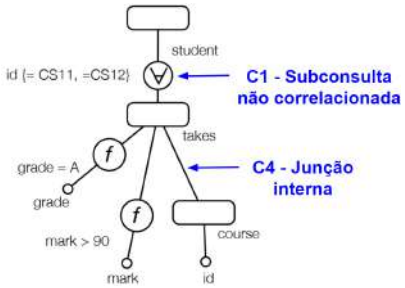
ícone. Em relação à PP2, a Figura 22 mostra o uso de subconsulta não correlacionada (C1) e junção interna (C4). Em relação à PP3, a Figura 22 mostra uma subconsulta na cláusula *Where* (C12). Em relação à PP4, a linguagem é suportada por uma ferramenta CASE, mas não está disponível para download. Em relação à PP5, o estudo apresenta uma avaliação experimental. Em resumo, esse estudo considera os critérios C1, C4 e C12.

Figura 21 – Exemplos de execução em DataPlay



Quem são os funcionários com salário superior ao salário médio geral?

Figura 22 – Principais recursos de DataPlay



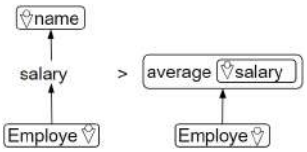
Quais alunos estão matriculados na classe A, recebem notas acima de 90 e fazem os cursos CS11 e CS12?

Fonte: adaptado de Abouzied, Hellerstein e Silberschatz (2012)

Kaleidoquery (A06)

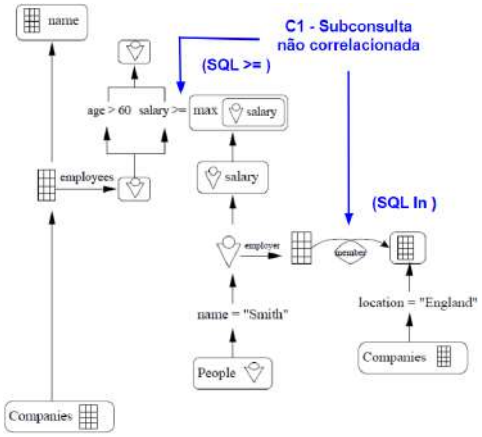
A Kaleidoquery (MURRAY et al., 2000) permite a especificação de subconsultas não correlacionadas (C1) (cf. Figura 23), mas, mesmo usando um construtor específico para subconsulta (e.g., o sinal maior que entre salário e salário médio), ele não pode ser usado para especificar subconsultas correlacionadas. Isso ocorre porque esse construtor é específico para subconsultas não correlacionadas, pois não permite uma condição de seleção com alias na consulta interna. Em relação à PP1, a Kaleidoquery é uma VQL baseada em diagrama e notação de ícone. Em relação à PP2, as figuras 24a, 24b, 24c e 24d mostram o uso de subconsulta correlacionada (C1), junção interna (C4), operação de conjunto (C6) e restrição de agrupamento (C7). Em relação à PP3, a Figura 24a mostra uma subconsulta na cláusula Where (C12). Em relação à PP4, a linguagem é suportada por uma ferramenta CASE, mas não está disponível para download. Em relação à PP5, o estudo apresenta uma avaliação experimental. Em resumo, esse estudo considera os critérios C1, C4, C6, C7 e C12.

Figura 23 – Exemplos de execução em Kaleidoquery

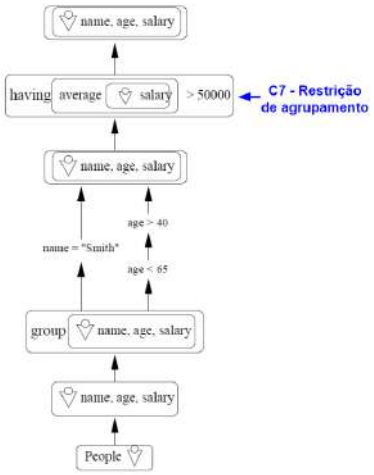


Quem são os funcionários com salário superior ao salário médio geral?

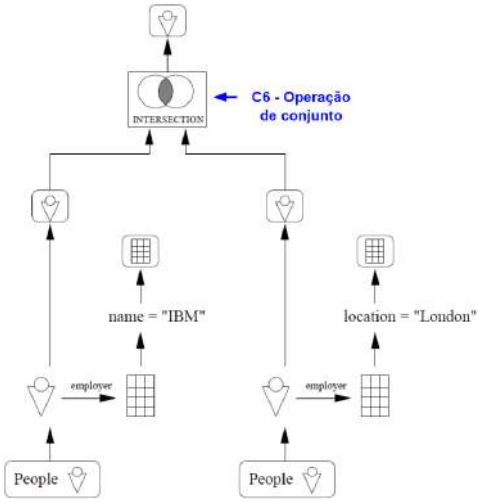
Figura 24 – Principais recursos de Kaleidoquery



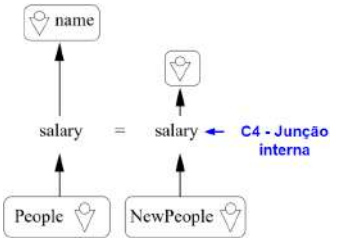
(a) Qual é o nome das pessoas com mais de 60 anos e com salário maior ou igual ao salário mais alto das pessoas chamadas Smith que trabalham em empresas localizadas na Inglaterra?



(b) Qual é o nome, a idade e o salário das pessoas nomeadas Smith com mais de 40 anos e menos de 65 anos, com salário médio superior a 50,000?



(c) Quais pessoas que trabalham na IBM e que também trabalham em empresas localizadas em Londres?



(d) Qual é o nome das pessoas que têm o mesmo salário que NewPeople?

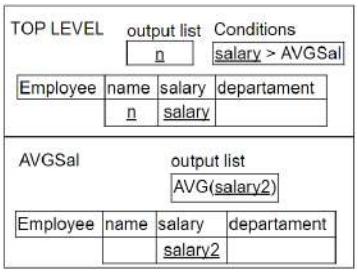
Fonte: adaptado de Murray et al. (2000)

Abe (A07)

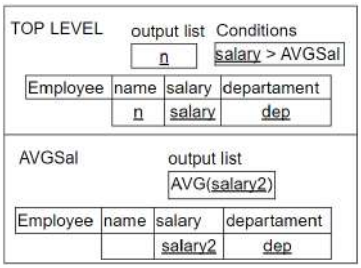
A Abe (KLUG, 1981) permite a especificação de subconsultas não correlacionadas (C1) e correlacionadas (C2) (cf. Figuras 25a e 25b). As subconsultas são definidas como blocos, com o

bloco de consulta externa nomeado com a palavra reservada TOP LEVEL e o bloco de consulta interna definido abaixo do bloco TOP LEVEL (i.e., o bloco AVGSal). Para especificação de subconsulta correlacionada, a Abe também usa alias (i.e., o atributo de *departament* como *dep*) e condições de seleção na consulta interna. Em relação à PP1, Abe é uma VQL baseada em notação de formulário. Em relação à PP2, as figuras 26a, 26b e 26c mostram o uso de subconsulta não correlacionada (C1), subconsulta correlacionada (C2), junção interna (C4) e restrição de agrupamento (C7). Em relação à PP3, as figuras 26a e 26b mostram uma subconsulta na cláusula Where (C12). Em relação à PP4, a linguagem é suportada por uma ferramenta CASE, mas não está disponível para download. Em relação à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo considera os critérios C1, C2, C4, C7 e C12.

Figura 25 – Exemplos de execução em Abe

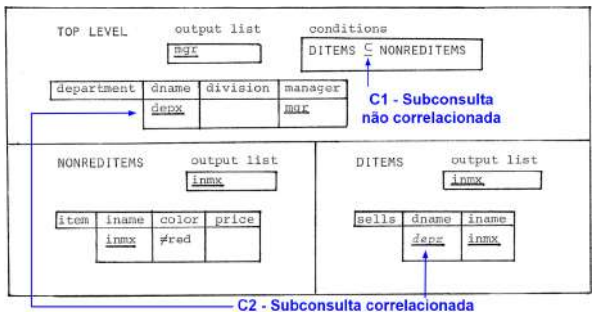


(a) Quem são os funcionários com salário superior ao salário médio geral?

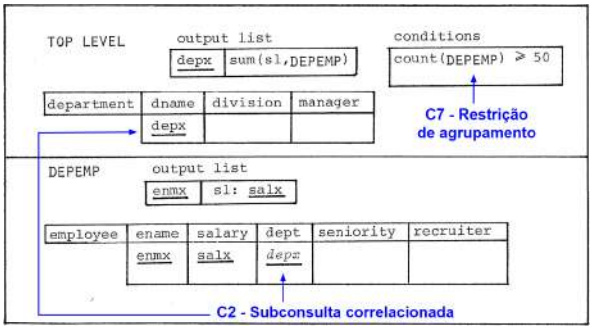


(b) Quem são os funcionários com salário superior ao salário médio de seu departamento?

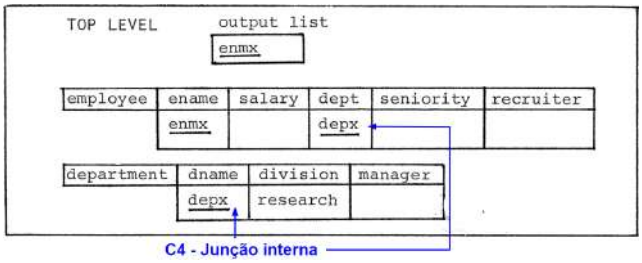
Figura 26 – Principais recursos de Abe



(a) Quem são os gerentes dos departamentos que não vendem itens vermelhos?



(b) Qual é a soma dos salários dos funcionários dos departamentos com pelo menos 50 funcionários?



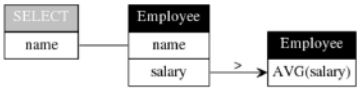
(c) Quem são os funcionários que trabalham nos departamentos da divisão de pesquisa?

Fonte: adaptado de Klug (1981)

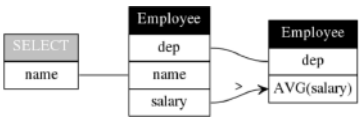
QueryViz (A08)

A QueryViz (DANAPARAMITA; GATTERBAUER, 2011) permite a especificação de subconsultas não correlacionadas (C1) e correlacionadas (C2) (cf. Figuras 27a e 27b). Essas subconsultas são definidas usando o mesmo construtor gráfico (i.e., uma seta sem um círculo - cf. a Figura 28a), mas a condição de correlação pode ser adicionalmente representada com outra ligação (cf. a Figura 27b). Ressalta-se que a QueryViz sobrecarrega a semântica do construtor de subconsultas, o que pode gerar interpretações ambíguas. Em relação à PP1, a QueryViz é uma VQL baseada na notação de diagrama. Em relação à PP2, as figuras 28a e 28b mostram o uso de subconsulta correlacionada (C2) e de junção interna (C4). Em relação à PP3, a Figura 28 mostra uma subconsulta na cláusula Where (C12). Em relação à PP4, a linguagem é suportada por uma ferramenta CASE disponível on-line [3]. Em relação à PP5, o estudo apresenta uma avaliação experimental. Em resumo, esse estudo inclui os critérios C1, C2, C4 e C12.

Figura 27 – Exemplos de execução em QueryViz

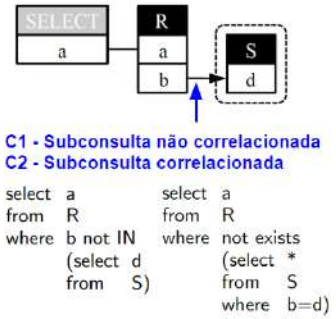


(a) Quem são os funcionários com salário superior ao salário médio geral?

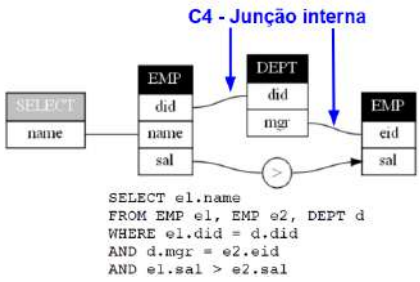


(b) Quem são os funcionários com salário superior ao salário médio de seu departamento?

Figura 28 – Principais recursos de QueryViz



(a) Quais são os "R" que não têm relação com "S"?



(b) Quem são os funcionários que ganham mais do que o gerente?

Fonte: adaptado de Danaparamita e Gatterbauer (2011)

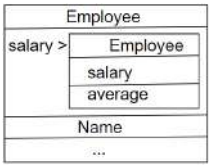
TableTalk (A09)

A TableTalk (EPSTEIN, 1991) permite a especificação de subconsultas não correlacionadas (C1) (cf. Figura 29), mas, mesmo usando um construtor específico para subconsulta, ela não pode ser usada para especificar subconsultas correlacionadas. Isso ocorre porque esse construtor não permite a especificação de uma condição de seleção com alias na consulta interna. Em

[3] http://queryviz.com/online

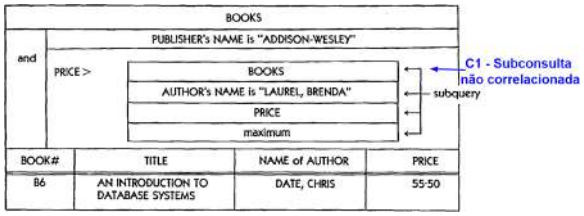
relação à PP1, a TableTalk é uma VQL baseada em notação de formulário. Em relação à PP2, as figuras 30a e 30b mostram o uso de subconsulta não correlacionada (C1) e junção interna (C4). Em relação à PP3, a Figura 30a mostra uma subconsulta na cláusula Where (C12). Em relação à PP4, a linguagem é suportada por uma ferramenta CASE, mas não está disponível para download. Em relação à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo considera os critérios C1, C4 e C12.

Figura 29 – Exemplos de execução em TableTalk

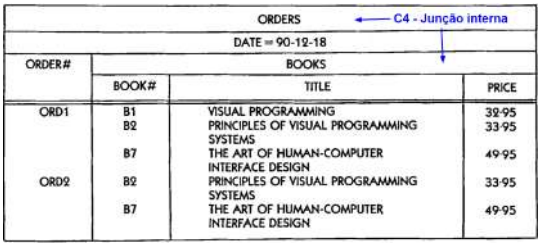


Quem são os funcionários com salário superior ao salário médio geral?

Figura 30 – Principais recursos de TableTalk



(a) Quais são os números, títulos, autores e preços dos livros publicados por Addison-Wesley que são mais caros que o livro mais caro escrito por Brenda Laurel?



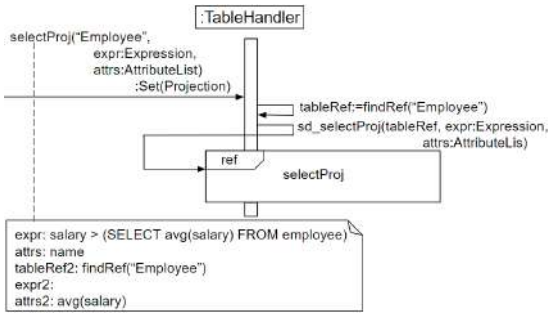
(b) Quais são os números de pedidos, números de livros, títulos e preços de todos os livros encomendados em 18 de dezembro de 1990?

Fonte: adaptado de Epstein (1991)

Song et al. (A10)

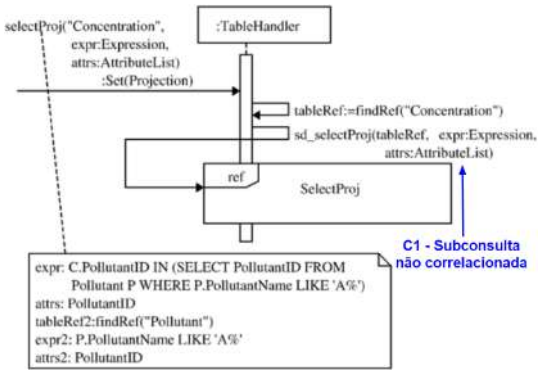
A linguagem proposta por Song et al. (SONG; YIN; RAY, 2007) permite a especificação de subconsultas não correlacionadas (C1) (cf. Figura 31), mas, mesmo usando um construtor específico para subconsulta (e.g., o diagrama de sequência *selectProj*, no qual a lógica para especificar uma subconsulta é definida), não pode ser usada para especificar subconsultas correlacionadas. Isso ocorre porque esse construtor não permite uma condição de seleção com alias na consulta interna. Em relação à PP1, essa linguagem é uma VQL baseada em notação de diagrama. Em relação à PP2, as figuras 32a, 32b, 32c e 32d mostram o uso de subconsulta correlacionada (C1), junção cruzada (C3), junção interna (C4) e operação de conjunto (C6). Em relação à PP3, a Figura 32b mostra uma subconsulta na cláusula Where (C12). Em relação à PP4, as consultas podem ser modeladas usando qualquer ferramenta UML CASE, mas nenhum código SQL pode ser gerado ou validado. Em relação à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo inclui os critérios C1, C3, C4, C6 e C12.

Figura 31 – Exemplos de execução na linguagem proposta por Song et al.

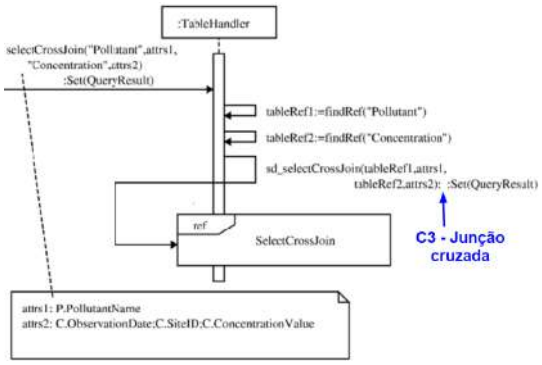


Quem são os funcionários com salário superior ao salário médio geral?

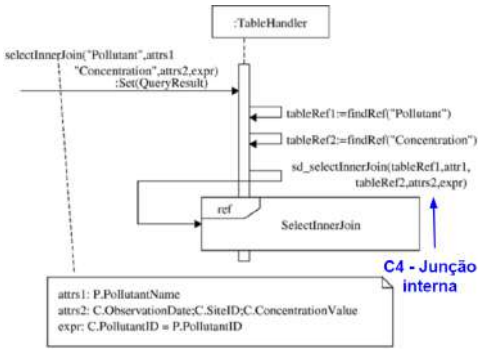
Figura 32 – Principais recursos da linguagem proposta por Song et al.



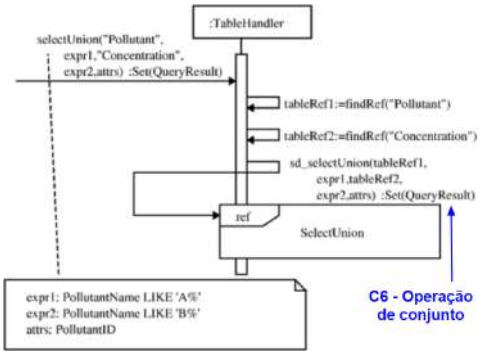
(a) Qual é a concentração de poluentes cujos nomes começam com A?



(b) Quais são os nomes dos poluentes, as datas de observação e os valores de concentração observados?



(c) Quais são os nomes, datas de observação e valores de concentração dos poluentes observados?



(d) Quais nomes de poluentes que começam com A juntamente com poluentes cujos nomes começam com B?

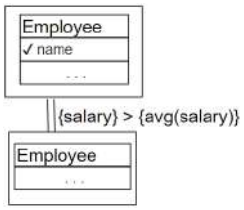
Fonte: adaptado de Song, Yin e Ray (2007)

Visual SQL (A11)

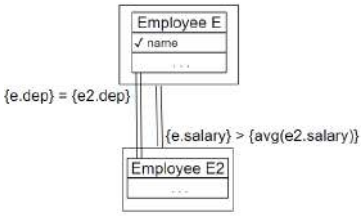
A Visual SQL (JAAKKOLA; THALHEIM, 2003) permite a especificação de subconsultas não correlacionadas (C1) e correlacionadas (C2) (cf. Figuras 33a e 33b). Ambos os tipos de subconsultas são definidos usando alias e um construtor específico para subconsultas (i.e., uma linha dupla entre consultas ou tabelas). Em uma subconsulta não correlacionada, um alias é usado para especificar uma condição de seleção (i.e, "salário $>$ avg (salário)") e a linha dupla

é especificada entre as consultas participantes. Em uma subconsulta correlacionada, um alias é usado para especificar uma condição de seleção na consulta interna (i.e., "e.dep > e2.dep") e a linha dupla é especificada entre as tabelas participantes da seleção condição. Em relação à PP1, a VisualSQL é uma VQL com notação baseada em diagrama e ícone. Em relação à PP2, a Figura 34 mostra o uso de subconsulta não correlacionada (C1), subconsulta correlacionada (C2) e junção interna (C4). Em relação à PP3, a Figura 34 mostra uma subconsulta na cláusula Where (C12). Em relação à PP4, a linguagem é suportada por uma ferramenta CASE, mas não está disponível para download. Em relação à PP5, o estudo apresenta uma avaliação experimental. Em resumo, esse estudo considera os critérios C1, C2, C4 e C12.

Figura 33 – Exemplos de execução em VisualSQL

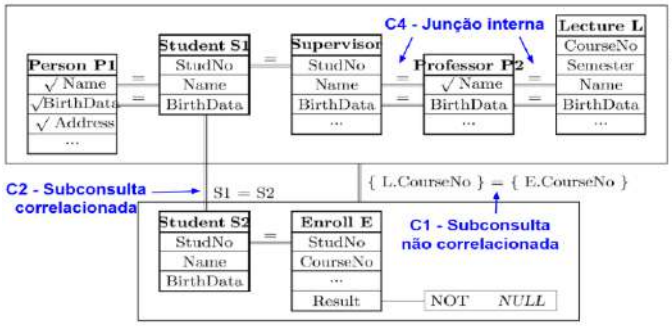


(a) Quem são os funcionários com salário superior ao salário médio geral?



(b) Quem são os funcionários com salário superior ao salário médio de seu departamento?

Figura 34 – Principais recursos de VisualSQL




```
SELECT P1.Name, P1.BirthData, P1. Address,
       P2.Name AS "Name of supervisor"
FROM Person P1, Professor P2, Student S1, Supervisor,
     Lecture L, Enroll E
WHERE P1.Name = S1.Name
    AND P1.BirthData = S1.BrirthData
    AND S1.StudNo = E.StudNo
    AND S1.Result IS NOT NULL
    AND S1.StudNo = Supervisor.StudNo
    AND Supervisor.Name = P2.Name
    AND Supervisor.BirthData = P2.BirthData
    AND L.Name = Professor.Name
    AND L.BirthData = P2.BirthData
    AND L.CourseNo IN ( SELECT E2.CourseNo
        FROM Enroll E2
        WHERE S1.StudNo = E2.StudNo
            AND E2.Result IS NOT NULL)
    AND E.CourseNo IN ( SELECT L2.CourseNo
        FROM Lecture L2
        WHERE L2.Name = P2.Name
        AND L2.BirthData = L2.BirthData);
```

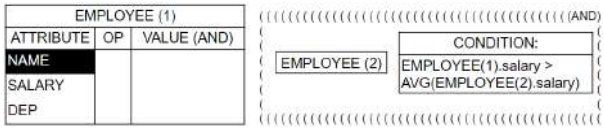
Quais alunos concluíram com êxito apenas os cursos que foram ou atualmente são administrados pelo seu supervisor?

Fonte: adaptado de Jaakkola e Thalheim (2003)

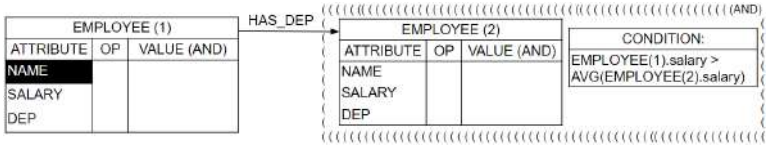
GRAQULA (A12)

A GRAQULA (SOCKUT et al., 1993) permite a especificação de subconsultas não correlacionadas (C1) e correlacionadas (C2) (cf. Figuras 35a e 35b). Ambos os tipos de subconsultas são definidos usando alias e um construtor de subconsultas específico (i.e., um bloco com as margens representadas pelo símbolo de parênteses). Em uma subconsulta não correlacionada, um alias é usado para especificar uma condição de seleção (i.e., "EMPLOYEE(1).salary >

Figura 35 – Exemplos de execução em GRAQULA



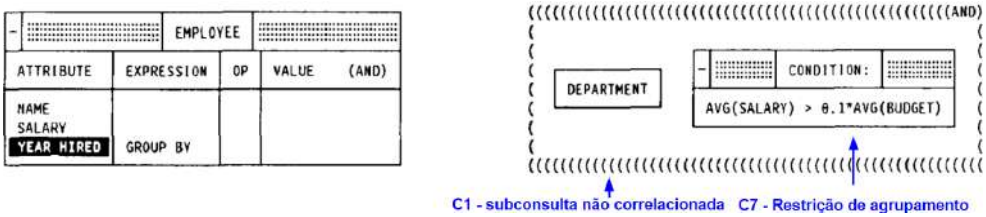
(a) Quem são os funcionários com salário superior ao salário médio geral?



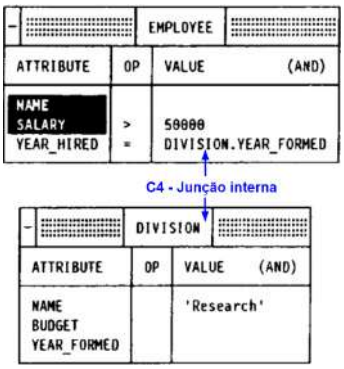
(b) Quem são os funcionários com salário superior ao salário médio de seu departamento?

Figura 36 – Principais recursos de GRAQULA

(a) Quais são os nomes dos funcionários que não possuem o título de programador e a habilidade de programação em C?



(b) Quais são os anos de trabalho para os quais o salário médio dos funcionários contratados naquele ano excede um décimo do orçamento médio de todos os departamentos?



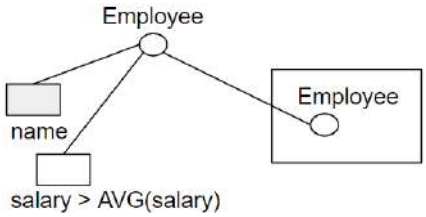
(c) Qual é o nome e o salário dos funcionários da divisão de pesquisa que ganham mais de 50,000?

Fonte: adaptado de Sockut et al. (1993)

AVG(EMPLOYEE(2).salary")). Em uma subconsulta correlacionada, uma seta é usada para especificar uma condição de correlação implícita com a consulta interna (i.e., "HAS_DEP"). Ressalta-se que o rótulo da seta deve conter o nome das colunas que fazem parte da condição de seleção precedida por "HAS_". Em relação à PP1, GRAQULA é uma VQL baseada em

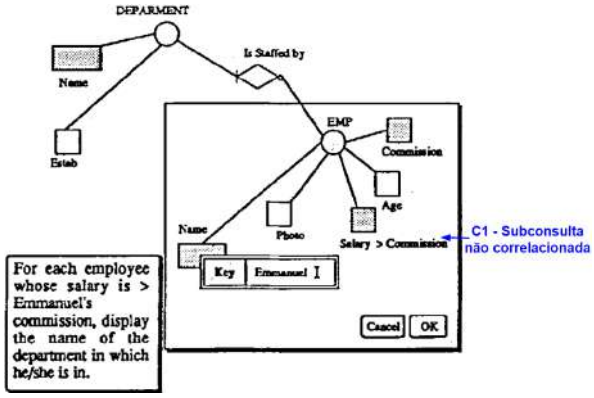
notação de diagrama e formulário. Em relação à PP2, as figuras 36a, 36b e 36c mostram o uso de subconsulta não correlacionada (C1), subconsulta correlacionada (C2), junção interna (C4) e restrição de agrupamento (C7). Em relação à PP3, as figuras 36a e 36b mostram uma subconsulta na cláusula Where (C12). Em relação à PP4, a linguagem não possui ferramenta CASE. Em relação à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo inclui os critérios C1, C2, C4, C7 e C12.

Figura 37 – Exemplos de execução em VILD

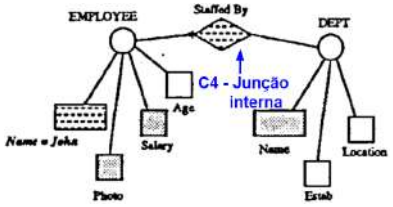


Quem são os funcionários com salário superior ao salário médio geral?

Figura 38 – Principais recursos de VILD



(a) Quais departamentos têm funcionários com um salário maior que a comissão de Emmanuel?



(b) Quais funcionários trabalham no mesmo departamento que John?

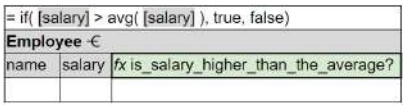
Fonte: adaptado de Mun-Kew et al. (1990)

VILD (A13)

A *Visual Language for Database* (VILD) (MUN-KEW et al., 1990) permite a especificação de subconsultas não correlacionadas (C1) (cf. Figura 37), mas mesmo usando um construtor específico para a subconsulta (i.e., um retângulo que destaca a subconsulta), ele não pode ser usado para especificar subconsultas correlacionadas, porque esse construtor não permite uma condição de seleção com alias na consulta interna. Em relação à PP1, a VILD é uma VQL baseada na notação de diagrama. Em relação à PP2, as figuras 38a e 38b mostram o uso de subconsulta não correlacionada (C1) e junção interna (C4). Ressalta-se que a VILD não permite subconsultas correlacionadas porque não suporta os dois recursos básicos para especificar uma correlação (i.e., uma condição de seleção com alias na consulta interna ou

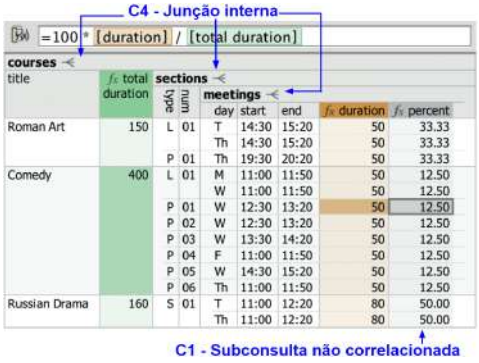
em um construtor específico - cf. Seção 3.7). Em relação à PP3, a Figura 38a mostra uma subconsulta na cláusula Where (C12). Em relação à PP4, a linguagem é suportada por uma ferramenta CASE, mas não está disponível para download. Em relação à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo considera os critérios C1, C4 e C12.

Figura 39 – Exemplo de execução em SIEUFERD

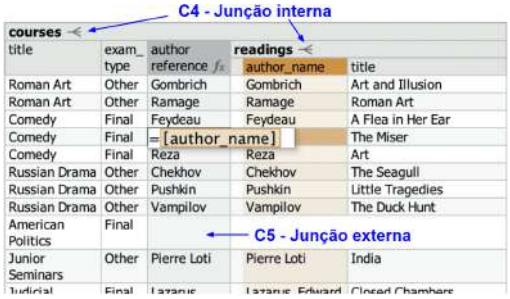


Quem são os funcionários com salário superior ao salário médio geral?

Figura 40 – Principais recursos de SIEUFERD



(a) Qual é a porcentagem de minutos para cada reunião de todas as reuniões realizadas durante o curso?



(b) Quais são as leituras recomendadas para cada curso? (Se o curso não tiver recomendações, ele deverá ser retornado na pesquisa)

(c) Quais são os departamentos por curso?

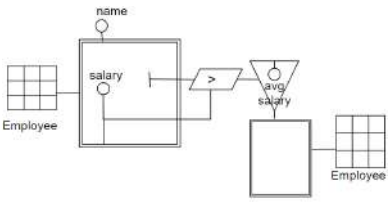
Fonte: adaptado de Bakke e Karger (2016)

SIEUFERD (A14)

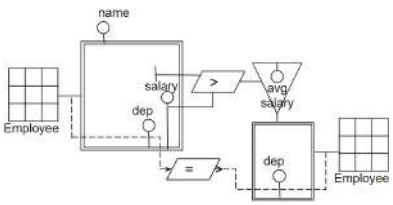
A SIEUFERD (BAKKE; KARGER, 2016) permite a especificação de subconsultas não correlacionadas (C1) (cf. Figura 39). Mesmo usando um construtor de subconsulta específico (i.e., fórmulas do tipo Excel que permitem especificar condições de seleção entre duas entidades), ela não suporta subconsultas correlacionadas porque não permite a especificação de uma condição de seleção com alias na consulta interna. Em relação à PP1, a SIEUFERD é uma VQL baseada em notação de formulário e ícone. Em relação à PP2, as figuras 40a, 40b e 40c mostram o uso da subconsulta correlacionada (C1), junção interna (C4), junção externa

(C5) e operação de conjunto (C6). Além das consultas apresentadas, os autores afirmam que o "produto cartesiano x é uma junção interna com C = true" (BAKKE; KARGER, 2016). Portanto, também é possível especificar uma junção cruzada (C3). Em relação à PP3, a Figura 40b mostra uma subconsulta na cláusula Select (C10). Em relação a RQ4, a linguagem é suportada por uma ferramenta CASE, mas não está disponível para download. Em relação à PP5, o estudo apresenta uma avaliação experimental. Em resumo, esse estudo considera os critérios C1, C3, C4, C5, C6 e C10.

Figura 41 – Exemplos de execução em GraphSQL



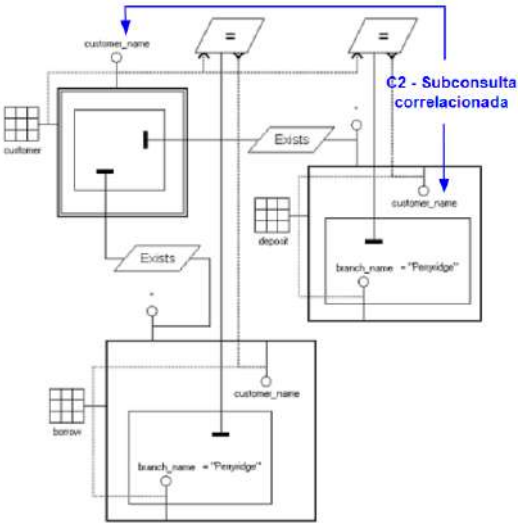
(a) Quem são os funcionários com salário superior ao salário médio geral?



(b) Quem são os funcionários com salário superior ao salário médio de seu departamento?

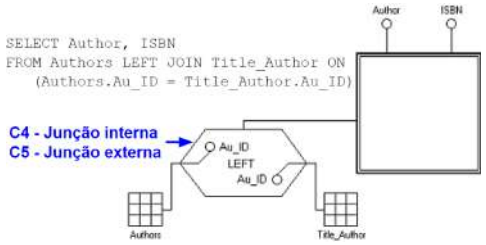
Figura 42 – Principais recursos de GraphSQL

```
SELECT C.customer_name
FROM customer C
WHERE (EXISTS (SELECT *
               FROM deposit D
               WHERE((C.customer_name=
                      D.customer_name) AND
                     (D.branch_name="Perryridge")))
  AND EXISTS (SELECT *
               FROM borrow B
               WHERE((B.branch_name="Perryridge") AND
               (C.customer_name=B.customer_name))))
```

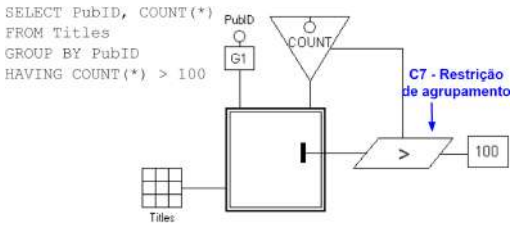


(a) Quais são os nomes dos clientes que depositam e tomam empréstimos de uma agência da Perryridge?

(b) Quais editores têm títulos publicados?



(c) Quais são os nomes e códigos ISBN de todos os autores que têm ou não uma publicação? (Se não forem publicados, o ISBN deve retornar nulo)



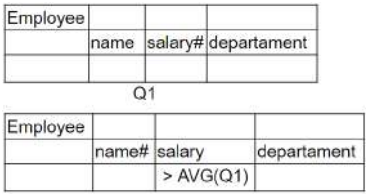
(d) Quais editores têm mais de 100 títulos publicados?

Fonte: adaptado de Cerullo e Porta (2007)

GraphSQL (A15)

A GraphSQL (CERULLO; PORTA, 2007) permite a especificação de subconsultas não correlacionadas (C1) e correlacionadas (C2) (cf. Figuras 41a e 41b). Os dois tipos de subconsultas são definidos usando um construtor de subconsulta específico. Em uma subconsulta não correlacionada, um paralelogramo é usado para definir uma condição de seleção entre duas consultas (i.e., "salário $>$ avg(salário)"). Na subconsulta correlacionada, um paralelogramo é usado para definir uma condição de seleção na consulta interna (i.e., "dep $=$ dep"). Em relação à PP1, a GraphSQL é uma VQL com notação baseada em diagrama e ícone. Em relação à PP2, as figuras 42a, 42b, 42c e 42d mostram o uso de subconsulta não correlacionados (C1), subconsulta correlacionada (C2), junção interna (C4), junção externa (C5) e restrição de agrupamento (C7). Além disso, os autores afirmam que a GraphSQL também possui outros símbolos usados para expressar vários aspectos da álgebra relacional (e.g., interseção e união entre os resultados de diferentes consultas) (CERULLO; PORTA, 2007). Portanto, também é possível especificar operações de conjunto (C6). Em relação à PP3, as figuras 42a e 42b mostram uma subconsulta na cláusula Where (C12). As subconsultas na cláusula Having são possíveis (C14) porque o operador de comparação apresentado na Figura 42d também aceita uma subconsulta como argumento (cf. Figura 42a). Em relação à PP4, a linguagem é suportado pela ferramenta GraphSQL Builder, mas não está disponível para download. Em relação à PP5, o estudo apresenta uma avaliação experimental. Em resumo, esse estudo considera os critérios C1, C2, C4, C5, C6, C7, C12 e C14.

Figura 43 – Exemplo de execução em GKQL



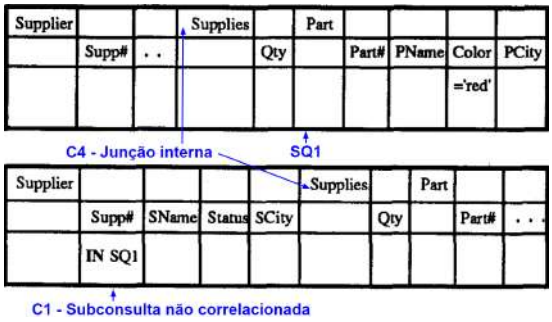
Quem são os funcionários com salário superior ao salário médio geral?

GKQL (A16)

A *Graphical Knowledge Level Query Language* (GKQL) (TAN; CHAN; SIAU, 1990) permite a especificação de subconsultas não correlacionadas (C1) (cf. Figura 43). Mesmo usando um alias (e.g., "salary $>$ AVG(Q1)"), ele não pode ser usado para referenciar atributos de uma relação, apenas atributos presentes na projeção da consulta. Em outras palavras, ele não permite uma condição de seleção com alias na consulta interna. Em relação à PP1, a GKQL é uma VQL baseada em notação de formulário. Em relação à PP2, a Figura 44 mostra o uso de subconsulta não correlacionada (C1) e de junção interna (C4). Em relação à PP3, a Figura 44 mostra uma subconsulta na cláusula Where (C12). Em relação à PP4, a linguagem é suportada por uma ferramenta CASE, mas não está disponível para download. Em relação

à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo considera os critérios C1, C4 e C12.

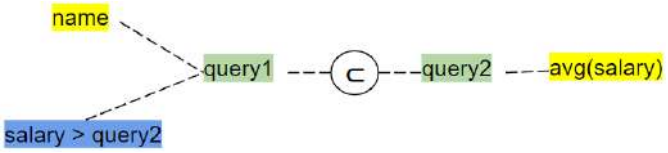
Figura 44 – Principais recursos de GKQL



Quais fornecedores fornecem pelo menos uma peça também fornecida pelo fornecedor de peças vermelhas?

Fonte: adaptado de Tan, Chan e Siau (1990)

Figura 45 – Exemplo de execução em Visque



Quem são os funcionários com salário superior ao salário médio geral?

Figura 46 – Principais recursos de Visque

Quantos filmes são alugados no primeiro trimestre, agrupados por trimestre e ano?

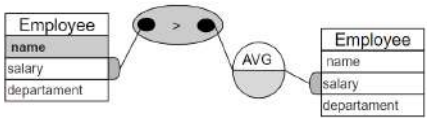
Fonte: adaptado de Borges e Macías (2010)

Visque (A17)

A Visque (BORGES; MACÍAS, 2010) permite a especificação de subconsultas não correlacionadas (C1) (cf. Figura 45), mas, mesmo usando um alias (e.g., "salary > query2"), não

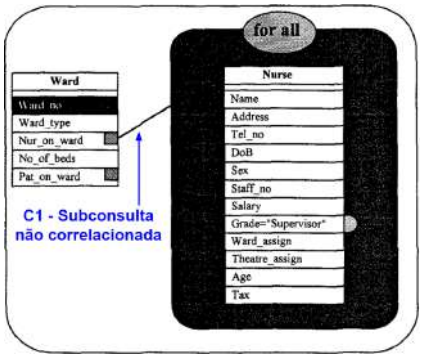
permite subconsultas correlacionadas. Isso ocorre porque o alias não pode ser usado para referenciar atributos de uma relação, apenas atributos presentes na projeção da consulta. Em relação à PP1, a Visque é uma VQL com notação baseada em diagrama e formulário. Em relação à PP2, a Figura 46 mostra o uso de subconsulta não correlacionada (C1) e operação de conjunto (C6). Em relação à PP3, a Figura 46 mostra uma subconsulta na cláusula Where (C12). Em relação à PP4, a linguagem é suportada por uma ferramenta CASE, mas não está disponível para download. Em relação à PP5, o estudo apresenta uma avaliação de pesquisa. Em resumo, esse estudo inclui os critérios C1, C6 e C12.

Figura 47 – Exemplo de execução em GOQL

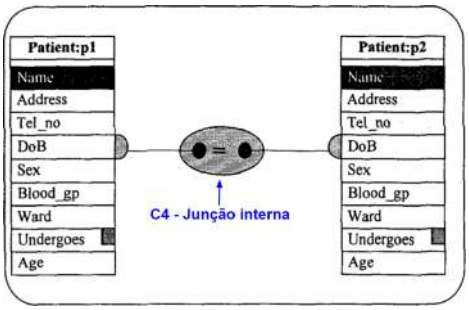


Quem são os funcionários com salário superior ao salário médio geral?

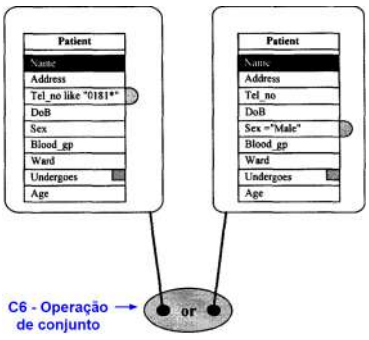
Figura 48 – Principais recursos de GOQL



(a) Quais são os números das enfermarias onde apenas os enfermeiros supervisionam?



(b) Quais são os nomes dos pacientes que têm a mesma data de nascimento?



(c) Quais são os nomes dos pacientes do sexo masculino ou cujo número de telefone começa com 0181?

Fonte: adaptado de Keramopoulos, Pouyioutas e Sadler (1997)

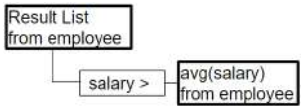
GOQL (A18)

A *Graphical Object-Oriented Query Language* (GOQL) (KERAMOPOULOS; POUYIOUTAS; SADLER, 1997) permite a especificação de subconsulta não correlacionada (C1) (cf. Figura 47). Mesmo usando um construtor de subconsulta não correlacionada específico (i.e., uma elipse em que a condição de seleção é definida conectada a uma função de agregação), ela não pode ser usada para especificar subconsultas correlacionadas, porque esse construtor não permite uma condição de seleção com alias na consulta interna. Ressalta-se que a GOQL também possui o construtor "for all" para projetar uma subconsulta (cf. Figura 48a). Em relação à PP1, a GOQL é uma VQL baseada na notação de diagrama. Em relação à PP2, as figuras 48a, 48b e 48c mostram o uso de subconsulta não correlacionada (C1); junção interna (C4) e operação de conjunto (C6). Em relação à PP3, a Figura 48a mostra uma subconsulta na cláusula Where (C12). Em relação à PP4, a linguagem não possui ferramenta CASE. Em relação à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo inclui os critérios C1, C4, C6 e C12.

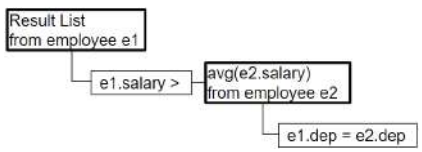
Keim e Lum (A19)

A linguagem proposta por Keim e Lum (KEIM; LUM, 1992) permite a especificação de subconsultas não correlacionadas (C1) e correlacionadas (C2) (cf. Figuras 49a e 49b). As subconsultas não correlacionados são definidas usando um construtor específico (i.e., o mesmo construtor usado para consulta). Em uma subconsulta correlacionada, um alias é usado para especificar uma condição de seleção na consulta interna (i.e., "e1.dep = e2.dep"). Em relação à PP1, essa linguagem é uma VQL baseada em notação de diagrama. Em relação à PP2, as figuras 50a e 50b mostram o uso de subconsulta não correlacionada (C1), subconsulta correlacionada (C2) e junção interna (C4). Além das consultas apresentadas, os autores afirmam que as funções estão divididas em cinco grupos: operadores lógicos, operadores de comparação, operadores de aninhamento, operadores de conjunto e operadores de agregação (KEIM; LUM, 1992). Portanto, também é possível especificar operações de conjunto (C6). Em relação à PP3, a Figura 50a mostra uma subconsulta na cláusula Where (C12). Em relação à PP4, a linguagem é suportada por uma ferramenta CASE, mas não está disponível para download. Em relação à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo considera os critérios C1, C2, C4, C6 e C12.

Figura 49 – Exemplos de execução na linguagem proposta por Keim e Lum

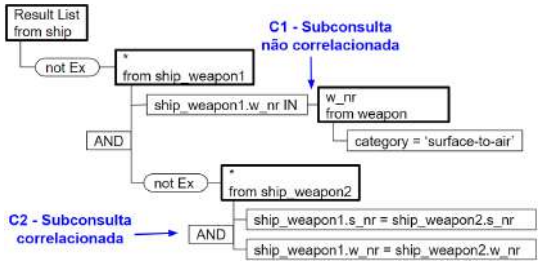


(a) Quem são os funcionários com salário superior ao salário médio geral?



(b) Quem são os funcionários com salário superior ao salário médio de seu departamento?

Figura 50 – Principais recursos da linguagem proposta por Keim e Lum

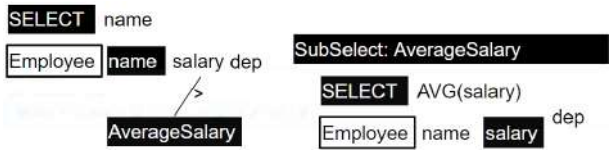


(a) Quais navios carregam armas exclusivas da categoria superfície-ar que nenhum outro navio carrega?

(b) Quais navios carregam armas?

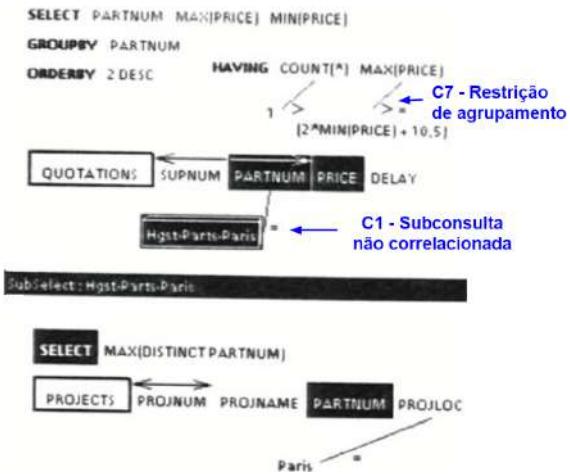
Fonte: adaptado de Keim e Lum (1992)

Figura 51 – Exemplos de execução em IQL

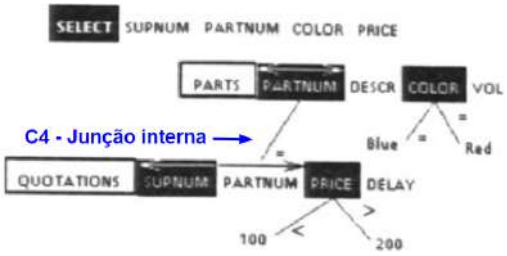


Quem são os funcionários com salário superior ao salário médio geral?

Figura 52 – Principais recursos de IQL



(a) Qual é o número de série, o preço máximo e o preço mínimo das cotações com: 1) número de série igual ao número de série mais alto das peças usadas pelo projeto pr1002 ou por projetos não alocados em Paris; 2) no máximo duas citações; e 3) preço máximo superior a duas vezes o preço mínimo mais 10,5?



(b) Quais são os números de fornecedores, números de peças, cor e preço das peças com: 1) peças vermelhas e preço inferior a 100; ou 2) peças azuis e preço não superior a 200?

Fonte: adaptado de Ramos (1993)

IQL (A20)

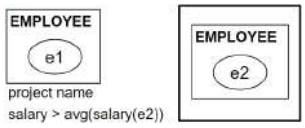
A *Interactive Query Language* (IQL) (RAMOS, 1993) permite a especificação de subconsulta não correlacionada (C1) (cf. Figura 51). Mesmo usando um construtor de subconsulta específico (e.g., "SubSelect: AverageSalary"), ele não pode ser usado para especificar subcon-

sultas correlacionadas, porque esse construtor não permite uma condição de seleção com alias na consulta interna. Em relação à PP1, a IQL é uma VQL baseada em notação de diagrama e formulário. Em relação à PP2, as figuras 52a e 52b mostram o uso de subconsulta não correlacionada (C1), restrição de agrupamento (C7) e junção interna (C4). Em relação à PP3, a Figura 52a mostra uma subconsulta na cláusula Where (C12). Em relação à PP4, a linguagem é suportada por uma ferramenta CASE, mas não está disponível para download. Em relação à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo inclui os critérios C1, C4, C7 e C12.

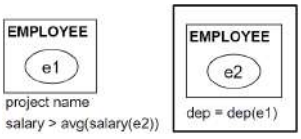
OQD (A21)

A *Object Query Diagram* (OQD) (KERAMOPOULOS; POUYIOUTAS; SADLER, 1997) permite a especificação de subconsultas não correlacionadas (C1) e correlacionadas (C2) (cf. Figuras 53a e 53b). Subconsultas não correlacionadas são definidas usando alias (i.e., "salary $>$ avg(salary(e2))"). Em uma subconsulta correlacionada, um alias também é usado para especificar uma condição de seleção na consulta interna (i.e., "dep $=$ dep(e1)"). Em relação à PP1, a OQD é uma VQL baseada na notação de diagrama. Em relação à PP2, as figuras 54a e 54b mostram o uso de subconsulta não correlacionada (C1), junção interna (C4) e junção externa (C5). Observe que a Figura 53b mostra o uso de uma subconsulta correlacionada (C2). Em relação à PP3, a Figura 54a mostra uma subconsulta na cláusula Where (C13). Em relação à PP4, a linguagem não possui ferramenta CASE. Em relação à PP5, o estudo não apresenta uma avaliação. Em resumo, esse estudo inclui os critérios C1, C2, C4, C5 e C12.

Figura 53 – Exemplos de execução em OQD

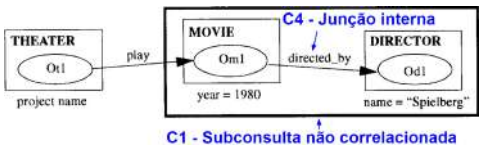


(a) Quem são os funcionários com salário superior ao salário médio geral?

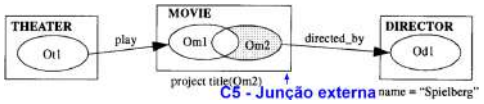


(b) Quem são os funcionários com salário superior ao salário médio de seu departamento?

Figura 54 – Principais recursos de OQD



(a) Quais teatros apresentaram todos os filmes dirigidos por Spielberg em 1980?



(b) Quais teatros apresentaram filmes dirigidos por Spielberg?

Fonte: adaptado de Keramopoulos, Pouyioutas e Sadler (1997)

3.7.1 Visão geral dos artigos selecionados

A Tabela 3 apresenta a expressividade de cada VQL, mostrando os critérios observados pela leitura dos artigos. Nessa tabela, um círculo completamente vazio indica que o critério não foi observado no artigo (i.e., não se pode afirmar que o critério é suportado) e um círculo completamente preenchido indica que o critério foi observado no artigo (i.e., foi encontrada uma evidência mostrando que o critério é suportado pelo trabalho). Na próxima seção, esses resultados são analisados e discutidos para responder às perguntas de pesquisa dessa revisão.

	PP2									PP3								
#	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18
A01	●	●	○	●	●	○	●	○	○	○	○	●	○	○	○	○	○	○
A02	●	●	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○
A03	●	●	○	●	○	○	●	○	●	○	○	●	○	○	○	○	○	○
A04	●	●	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○
A05	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○
A06	●	○	○	●	○	●	●	○	○	○	○	●	○	○	○	○	○	○
A07	●	●	○	●	○	○	●	○	○	○	○	●	○	○	○	○	○	○
A08	●	●	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○
A09	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○
A10	●	○	●	●	○	●	○	○	○	○	○	●	○	○	○	○	○	○
A11	●	●	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○
A12	●	●	○	●	○	○	●	○	○	○	○	●	○	○	○	○	○	○
A13	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○
A14	●	○	●	●	●	●	○	○	○	●	○	○	○	○	○	○	○	○
A15	●	●	○	●	●	●	●	○	○	○	○	●	○	●	○	○	○	○
A16	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○
A17	●	○	○	○	○	●	○	○	○	○	○	●	○	○	○	○	○	○
A18	●	○	○	●	○	●	○	○	○	○	○	●	○	○	○	○	○	○
A19	●	●	○	●	○	●	○	○	○	○	○	●	○	○	○	○	○	○
A20	●	○	○	●	○	○	●	○	○	○	○	●	○	○	○	○	○	○
A21	●	●	○	●	●	○	○	○	○	○	○	●	○	○	○	○	○	○

○ - evidência não encontrada; ● - evidência encontrada.

C1 - subconsulta não correlacionada; C2 - subconsulta correlacionada; C3 - junção cruzada;
C4 - junção interna; C5 - junção externa; C6 - operação de conjunto;
C7 - restrição de agrupamento; C8 - expressão condicional; C9 - recursão;
C10 - subconsulta na Select; C11 - subconsulta na From; C12 - subconsulta na Where;
C13 - subconsulta na Order By; C14 - subconsulta na Having;
C15 - expressão condicional na Select; C16 - expressão condicional na Where;
C17 -expressão condicional na Order By; C18 - expressão condicional na Having;

Tabela 3 – Visão geral dos artigos selecionados

3.8 ANÁLISE E DISCUSSÃO

Esta seção apresenta a análise e discussão que responde às perguntas de pesquisa da Seção 3.1.

PP1: qual é a abstração visual usada por cada VQL?

Essa pergunta de pesquisa investigou o tipo de notação visual (i.e., diagrama, formulário, ícones ou híbrido) usada por cada proposta de VQL. Essa pergunta é importante porque mostra em qual abstração visual a comunidade VQL está focada.

Os resultados resumidos na Tabela 4 mostram que a notação do tipo diagrama é a mais utilizada (38,09%, 8 artigos), seguida por formulário (19,4%, 4 artigos). Ressalta-se que as três últimas linhas da Tabela 4 mostram artigos que usam abordagens híbridas, ou seja, no total existem 15, 9 e 6 VQL que são baseadas em diagrama, formulário ou ícone, respectivamente.

Abstração visual	Artigos	Quant.	%
Diagrama	A03, A04, A08, A10, A13, A18, A19, A21	8	38,09
Formulário	A02, A07, A09, A16	4	19,04
Diagrama e icone	A05, A06, A11, A15	4	19,04
Diagrama e Formulário	A12, A17, A20	3	14,28
Formulário e icone	A01, A14	2	9,52

Tabela 4 – Tipos de abstrações visuais usadas nos estudos de VQL

PP2: quais são os construtores complexos suportados por cada VQL?

Nessa pergunta de pesquisa, foi investigado o número de VQL que suporta cada construção complexa (i.e., subconsulta, junção, operação de conjunto, expressão condicional, restrição de agrupamento e recursão). Essa pergunta é importante porque, além de mostrar a frequência (comunalidade) de cada construtor complexo, também ajuda a determinar a expressividade de cada VQL.

Os resultados resumidos na Tabela 5 revelam que: subconsultas não correlacionadas estão disponíveis em todos os artigos (100,00%, 21 artigos); junções internas estão disponíveis em quase todos os artigos (95,23%, 20 artigos); subconsultas correlacionadas estão presentes em mais da metade dos artigos (52,38%, 11 artigos); operações de conjunto e restrições de agrupamento estão disponíveis em um terço dos artigos (33,33%, 7 artigos); junções externas (19,04%, 4 artigos), junções cruzadas (9,52%, 2 artigos) e recursão (4,76%, 1 artigo) estão presentes em poucos artigos; e nenhum artigo suporta expressões condicionais. É importante reforçar que apenas o critério observado no artigo (i.e., foi encontrada uma evidência de que o critério é suportado pelo trabalho) foi considerado.

A análise dos resultados mostra que a falta de suporte para expressões condicionais e o baixo suporte para recursão, junções cruzadas, junções externas, operações de conjunto, expressões

condicionais e restrições de agrupamento afetam negativamente a expressividade da VQL, porque: 1) a semântica desses construtores dificilmente é substituída por outras construções complexas; 2) as expressões condicionais são necessárias para criar consultas de negócios do mundo real (GRYZ et al., 2008); 3) o uso de recursão é obrigatório para especificar consultas com fecho transitivo (i.e., consultas que permitem encadeamento com base no valor anterior de um atributo) (EL-MAHGARY; SOISALON-SOININEN, 2015); 4) as junções cruzadas são úteis para obter um produto cartesiano; 5) existem consultas que só podem ser especificadas usando junções externas (e.g., obtenção de dados opcionais); e 6) as restrições de agrupamento são importantes para especificar consultas com condições em grupos resumidos.

Construção complexa	Artigos	Quant.	%
C1 - subconsulta não correlacionada	A01, A02, A03, A04, A05, A06, A07, A08, A09, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21	21	100,00
C2 - subconsulta correlacionada	A01, A02, A03, A04, A07, A08, A11, A12, A15, A19, A21	11	52,38
C3 - junção cruzada	A10, A14	2	9,52
C4 - junção interna	A01, A02, A03, A04, A05, A06, A07, A08, A09, A10, A11, A12, A13, A14, A15, A16, A18, A19, A20, A21	20	95,23
C5 - junção externa	A01, A14, A15, A21	4	19.04
C6 - operação de conjunto	A06, A10, A14, A15, A17, A18, A19	7	33,33
C7 - restrição de agrupamento	A01, A03, A06, A07, A12, A15, A20	7	33,33
C8 - expressão condicional	∅	0	0,00
C9 - recursão	A03	1	4,76

Tabela 5 – Frequência de cada construção complexa

PP3: qual é o suporte para especificar subconsultas ou expressões condicionais fornecidas por cada VQL?

Para essa pergunta de pesquisa foi investigado o número de VQL que permitem a especificação de uma subconsulta ou expressão condicional dentro das cláusulas SQL. Ou seja, cada VQL foi examinada para verificar se suporta a definição de subconsultas nas cláusulas Select, From, Where, Order by ou Having e a definição de expressões condicionais nas cláusulas Select, Where, Order by ou Having.

Os resultados resumidos na Tabela 6 mostram que quase todas as VQL permitem subconsultas na cláusula Where (95,23%, 20 artigos), apenas uma VQL permite subconsultas na cláusula Having (4,76%; 1 artigo), apenas uma VQL permite subconsultas a cláusula Select (4,76%, 1 artigo) e nenhuma VQL permite subconsultas nas cláusulas From ou Order By. Além disso, nenhuma VQL permite expressões condicionais nas cláusulas SQL.

Em resumo, os resultados mostram que nenhum artigo suporta expressões condicionais e, embora todos os artigos suportem subconsulta, quase todas as VQL suportam esse construtor

apenas na cláusula Where. A falta de suporte para especificar expressões condicionais nas cláusulas Select, Where, Having e Order By afeta negativamente a expressividade das VQL, porque requer o uso de procedimentos, tornando o código SQL proprietário de um sistema de gerenciamento de banco de dados específico. A falta de suporte para especificar subconsultas nas cláusulas Select, From, Having e Order By também afeta negativamente a expressividade das VQL porque: 1) as subconsultas na cláusula Select são úteis no cálculo de valores agregados (e.g., COUNT, AVG, MIN, MAX) de uma subconsulta na projeção da consulta principal (cf. Figura 6a); 2) as subconsultas na cláusula From permitem cálculos indiretos envolvendo mais de uma função agregada (cf. Figura 6c). Além disso, também é útil para criar consultas otimizadas, porque pode restringir linhas e colunas antes de executar um produto cartesiano ou junção (DARMAWIKARTA, 2016); 3) as subconsultas na cláusula Having são úteis no agrupamento de restrições para evitar a comparação com valores codificados (cf. Figura 6b). Ressalta-se que uma comparação com um valor agregado só pode ser especificada na cláusula Having, porque gerará um erro de sintaxe se especificado na cláusula Where; e 4) as subconsultas na cláusula Order By são úteis para determinar a ordem de classificação e os valores OFFSET ou FETCH das linhas com base em um determinado resultado da subconsulta.

Cláusula SQL	Artigos	Quant.	%
C10 - subconsulta na cláusula Select	A14	1	4,76
C11 - subconsulta na cláusula From	∅	0	0
C12 - subconsulta na cláusula Where	A01, A02, A03, A04, A05, A06, A07, A08, A09, A10, A11, A12, A13, A15, A16, A17, A18, A19, A20, A21	20	95,23
C13 - subconsulta na cláusula Order By	∅	0	0
C14 - subconsulta na cláusula Having	A15	1	4,76
C15 - expressão condicional na cláusula Select	∅	0	0
C16 - expressão condicional na cláusula Where	∅	0	0
C17 - expressão condicional na cláusula Order By	∅	0	0
C18 - expressão condicional na cláusula Having	∅	0	0

Tabela 6 – Frequência de cada cláusula SQL no suporte a subconsultas ou expressões condicionais

PP4: qual é o suporte de ferramenta CASE fornecido por cada VQL?

Essa pergunta de pesquisa investigou se as ferramentas CASE das VQL estavam disponíveis, indisponíveis ou não existiam. Esse recurso é crítico para o uso prático das VQL. Ou seja, sem uma ferramenta CASE, é impossível detectar erros, executar engenharia reversa ou gerar artefatos de software, como a documentação do sistema.

Os resultados resumidos na Tabela 7 revelam que mais da metade das VQL (66,66%, 14 artigos) possui uma ferramenta que não está disponível para download, a maioria das demais

VQL restantes não possui uma ferramenta (23,80%, 5 artigos) e apenas duas propostas são suportadas por uma ferramenta CASE disponível para o usuário final (9,52%; 2 artigos).

Disponibilidade	Artigos	Quant.	%
Ferramenta CASE disponível	A08, A10	2	9,52
Ferramenta CASE indisponível	A01, A03, A05, A06, A07, A09, A11, A13, A14, A15, A16, A17, A19, A20	14	66,66
Não existe ferramenta CASE	A02, A04, A12, A18, A21	5	23,80

Tabela 7 – Disponibilidade da ferramenta CASE

A análise dos resultados mostra que a maioria das propostas de VQL supostamente desenvolveu uma ferramenta CASE, mas apenas a VQL dos artigos A08 e A10 é suportada por uma ferramenta CASE. A indisponibilidade das ferramentas CASE dificulta o uso das VQL na academia ou na indústria. No meio acadêmico, uma ferramenta CASE pode ser usada como um instrumento de avaliação empírica, bem como uma prova de conceito mostrando a viabilidade da VQL. Na indústria, sem uma ferramenta CASE, não há razão para usar a VQL.

PP5: qual é o método usado para avaliar cada VQL?

Essa pergunta de pesquisa investigou os tipos de métodos de avaliação (e.g., *survey*, entrevista ou experimento) usados em cada proposta de VQL. Essa pergunta é importante para avaliar a maturidade do campo de pesquisa.

Método de avaliação	Artigos	Quant.	%
Survey	A17	1	4.76
Experimento	A05, A06, A08, A11, A14, A15	6	28.57
Não há avaliação	A01, A02, A03, A04, A07, A09, A10, A12, A13, A16, A18, A19, A20, A21	14	66.66

Tabela 8 – Tipos de métodos de avaliação usados nos estudos VQL

Os resultados resumidos na Tabela 8 revelam que mais da metade das VQL (66,66%, 14 artigos) não avaliou suas linguagens, a maioria das propostas restantes fez uma avaliação experimental (28,57%, 6 artigos) e apenas uma fez uma avaliação com *survey* (5,76%; 1 artigo).

A análise dos resultados mostra que um terço das VQL propostas usavam um método de avaliação. A falta de um método de avaliação na maioria das propostas de VQL pode afetar negativamente sua funcionalidade, confiabilidade ou usabilidade; pois uma linguagem que não foi avaliada pelos usuários finais pode não suportar requisitos importantes do usuário.

3.9 AMEAÇAS DE VALIDADE

Esta seção apresenta preocupações relacionadas a ameaças de validade para essa RSL. Ou seja, é discutida a possibilidade de que os resultados obtidos não estejam de acordo com a realidade. De acordo com Maxwell (2012), as ameaças à validade podem ser classificadas em quatro categorias:

- **Validade de construto**: essa ameaça está relacionada a cenários em que as conclusões obtidas são válidas, mas deve ser verificado se os fatores que levam a tais conclusões correspondem à realidade. Uma ameaça para validade de construto seriam os termos da expressão de pesquisa e sua organização na cadeia de pesquisa. É possível que a cadeia construída possa não estar incluindo um termo importante na pesquisa. Para reduzir essa ameaça, os termos de pesquisa foram projetados para se concentrar nos objetivos das perguntas de pesquisa e abranger o máximo de resultados possíveis (i.e., os termos plural e singular foram usados);
- **Validade interna**: essa ameaça refere-se a situações não planejadas que podem ter ocorrido durante o processo de revisão e, para reduzir essa ameaça, os estudos incluídos foram revisados por pares;
- **Validade externa**: essa ameaça está relacionada à generalização dos resultados da pesquisa e, para reduzir essa ameaça, foram realizadas validações nos termos de pesquisa utilizados. Ou seja, a busca foi realizada e os resultados avaliados para verificar se eles satisfaziam os objetivos;
- **Validade de conclusão**: essa ameaça refere-se à verificação da aleatoriedade nos resultados obtidos em relação ao cenário planejado e, para reduzir essa ameaça, os critérios de inclusão, exclusão e qualidade foram planejados e aplicados com a assistência de especialistas.

3.10 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Nessa seção apresenta-se os resultados de uma RSL que investiga as limitações das VQL usadas para especificar consultas SQL complexas. Nessa revisão, a busca foi feita automaticamente e a avaliação foi realizada utilizando critérios de inclusão, exclusão e qualidade. Inicialmente, foi apresentada uma visão geral dos artigos selecionados. Em seguida, foram investigadas a abstração visual (PP1), a expressividade (PP2 e PP3), a disponibilidade das ferramentas CASE (PP4) e o método de avaliação (PP5) das VQL para especificar consultas SQL complexas. Por fim, foi apresentada uma discussão dos resultados. Foram analisados 21 artigos que propunham uma VQL, e esses forneceram as evidências para responder às cinco perguntas da pesquisa. As respostas das perguntas da pesquisa estão resumidas a seguir.

PP1: qual é a abstração visual usada por cada VQL? Verificou-se que diagrama é a abstração visual na qual a comunidade VQL está mais focada.

PP2: quais são os construtores complexos suportados por cada VQL? Verificou-se que as construções de junção interna e subconsulta estão disponíveis na maioria dos artigos; operação de conjunto, junção externa, junção cruzada e recursão estão presentes em alguns artigos; e nenhum artigo suporta expressões condicionais. É importante destacar que a falta de construtores para expressões condicionais e um suporte insuficiente para recursão, junções cruzadas, junções externas, operações de conjunto e restrições de agrupamento afetaram negativamente a expressividade do VQL.

PP3: qual é o suporte para especificar subconsultas ou expressões condicionais fornecidas por cada VQL? Como nenhuma VQL permite expressões condicionais, esse construtor não pode ser especificado em nenhuma cláusula SQL. Por sua vez, verificou-se que quase todas as VQL suportam apenas subconsultas na cláusula Where. Ressalta-se que a falta de suporte para especificar expressões condicionais em qualquer cláusula SQL, bem como subconsultas nas cláusulas Select, From, Having e Order By, também afetaram negativamente a expressividade da VQL.

PP4: qual é o suporte de ferramenta CASE fornecido por cada VQL? Apenas duas propostas são suportadas por uma ferramenta CASE disponível para o usuário final e mais da metade das VQL possuía uma ferramenta que não estava disponível para download.

PP5: qual é o método usado para avaliar cada VQL? Verificou-se que um terço das propostas de VQL utilizavam um método de avaliação, sendo a avaliação experimental o método mais utilizado.

Com base nos resultados, pode afirmar que um dos motivos pelos quais as VQLs não são amplamente utilizadas para especificar consultas complexas são: a baixa expressividade das VQL, o baixo suporte das ferramentas CASE aos usuários finais e a falta de avaliações rigorosas. Ressalta-se que as iniciativas não relacionais (e.g., Spark e Spanner) oferecem suporte ao SQL, mostrando a importância e o potencial dessa linguagem no contexto de *big data*. Além disso, destaca-se que VQL importantes (e.g., QBE (ZLOOF et al., 1975), QBD (SANTUCCI; SOTTILE, 1993) e QBI (BADRE et al., 1996)) não foram incluídas nessa RSL porque foram eliminadas pelo critério de inclusão IC2 (i.e., artigos que abordam a especificação de subconsulta).

No próximo capítulo apresenta-se a proposta desse trabalho, uma VQL denominada DSQL, que tem como objetivo superar as deficiências identificadas nessa RSL.

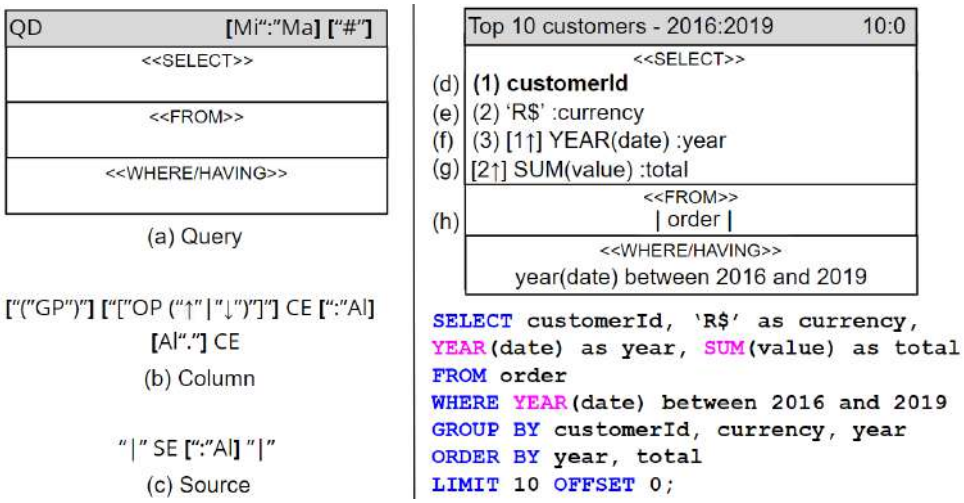
4 DIAGRAMMATIC STRUCTURED QUERY LANGUAGE

Com o objetivo de superar as limitações identificadas na RSL (cf. Capítulo 3), neste capítulo é proposta uma VQL denominada *Diagrammatic Structured Query Language* (DSQL), a qual permite especificar consultas SQL tanto simples quanto complexas (cf. Seção 2.1). No contexto do paradigma de desenvolvimento dirigido a modelos (i.e., MDD - cf. Seção 2.2), a DSQL é uma linguagem de modelagem específica de domínio (i.e., DSML). Isto é, uma linguagem diagramática com sintaxe concreta (i.e., notação) e sintaxe abstrata (i.e., metamodelo). Neste sentido, as seções deste capítulo estão organizadas da seguinte maneira: inicialmente, a VQL proposta é especificada a partir da descrição de suas sintaxes concreta e abstrata; na sequência, apresenta-se, como uma prova de conceito, a implementação de uma ferramenta CASE que implementa um conjunto inicial de regras de boa formação estrutural (i.e., semântica estática) e dá suporte a geração de código SQL a partir de diagramas DSQL; por fim, apresentam-se as considerações finais deste capítulo.

4.1 SINTAXE CONCRETA

O desenvolvimento da DSQL tem como ponto de partida a especificação da sua sintaxe concreta. Considerando o amplo uso e aceitação da notação do diagrama de classes UML (i.e., caixas, compartimentos e ligações), a sintaxe concreta da DSQL é baseada em uma representação diagramática semelhante. A Figura 55 apresenta os construtores básicos de DSQL (i.e., Query, Column e Source).

Figura 55 – Construtores Query, Column e Source



O construtor Query (Figura 55-a) possui quatro compartimentos. O compartimento superior possui um rótulo formado pela descrição da consulta (QD), seu limite mínimo e máximo de resultados ([Mi":"Ma]) e se a consulta deve retornar valores distinto (["#"]). Com exceção

da descrição da consulta, as outras propriedades são opcionais. Os outros compartimentos, de cima para baixo, mapeiam os conceitos: Select, From e Where/Having. O construtor Query pode representar uma consulta (Figura 55) e uma subconsulta (Figura 58).

O construtor Column (Figura 55-b) pode ser usado no compartimento Select ou no compartimento Where/Having. Em ambos compartimentos, este construtor é um elemento de coluna (CE) que pode representar: um campo de uma tabela (fonte em negrito - Figura 55-d); uma expressão livre (fonte normal - Figuras 55-e, f g); uma expressão condicional (Figura 57) ou uma subconsulta (Figura 58). Quando o construtor Column é usado no compartimento Select, este pode ter uma posição de agrupamento (["("GP")"]); uma posição de ordem (OP); um tipo de ordenação ($\uparrow$ = ascendente e $\downarrow$ = decrescente); e um elemento de coluna mais um alias (CE[":"Al]). Quando é usado no compartimento Where/Having, este apenas pode ter um alias mais um elemento de coluna ([Al"."] CE). Ressalta-se que: (1) com exceção do elemento de coluna (CE), as outras propriedades são opcionais; (2) a ordem de projeção segue a disposição especificada (de cima para baixo) no compartimento Select; e (3) a ordem do Group By e Order By são definidas pela numeração especificada entre parênteses e colchetes, respectivamente; (4) uma expressão livre (texto digitado pelo usuário) deve seguir as regras do SGBD e pode ser usada para escrever uma função SQL (e.g., LIKE, SUM ou YEAR), uma operação (e.g., aritmética, condicional ou lógica) ou literal (e.g., "R$"); e (5) Qualquer expressão livre no compartimento Where/Having que tenha uma função de agregação (e.g., SUM, AVG ou MIM) é a uma restrição de agrupamento (Having) - para os demais casos tem-se uma restrição de seleção (Where).

O construtor Source (Figura 55-c) pode ser usado para representar uma tabela (Figura 55-h) ou uma subconsulta (Figura 58-c). Quando um elemento de origem (SE) é uma tabela, seu rótulo deve descrever o nome da tabela; caso contrário, seu rótulo fica vazio. Nos dois casos, um alias (Al) é opcional ("|"SE[":"Al]"|"). O lado direito da Figura 55 mostra uma consulta básica usando esses construtores DSQL e sua representação em SQL. Essa consulta responde à seguinte pergunta: *"Quem são os dez principais clientes, de 2016 a 2019, ordenados por ano e total comprado?"*

Quando uma expressão lógica possui muitas operações, um recurso visual para favorecer seu entendimento é modularizar suas operações por conectivos lógicos. Nesse sentido, são necessários três construtores: (1) LogicalConnective - este construtor indica um conectivo lógico a ser usado (Figura 56-a); (2) LogicalExpression - esse construtor reúne o conjunto de operações de conjunção ou disjunção (Figura 56-b); e (3) LogicalLink - este construtor mostra a sequência de execução dos conectivos lógicos (Figura 56-c). O construtor LogicalConnective possui apenas um compartimento cujo rótulo é semelhante ao rótulo do construtor Column. Isto é, com exceção da propriedade elemento de coluna (EC), que só existe no construtor Column, as demais propriedades desses construtores são iguais. Vale ressaltar que estas propriedades apenas são exigidas quando é necessário especificar condições lógicas no compartimento Select, ou seja, estas propriedades não são necessárias no compartimento Where (a maioria dos casos).

O construtor LogicalExpression possui dois compartimentos: o superior é para especificar se a expressão lógica é um conjunto de conjunções ou disjunções, bem como suas negações (("AND" | "OR")["!"]). O inferior reúne o conjunto de operações lógicas. O lado direito da Figura 56 mostra uma consulta usando esses construtores DSQL e sua representação em SQL. Essa consulta responde à seguinte pergunta: *"Quem são os clientes que fizeram pagamentos à vista (dinheiro ou cartão de débito), entre 2016 e 2019, com um valor total do pedido maior que 10,000?"*

Figura 56 – Construtores para conectores lógicos

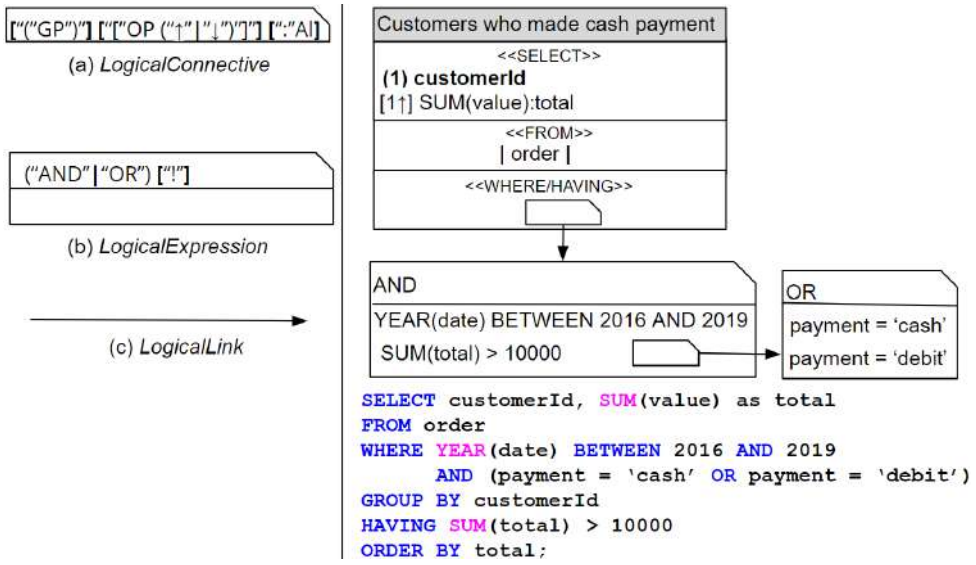
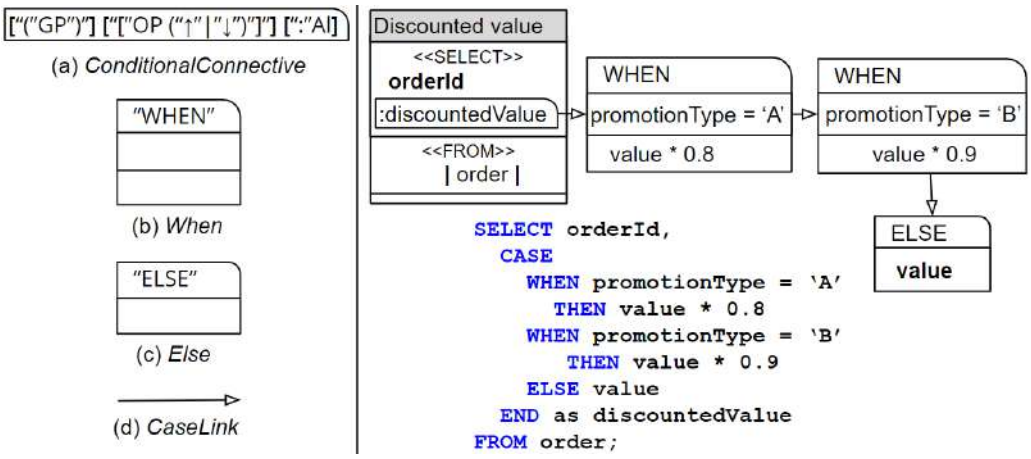


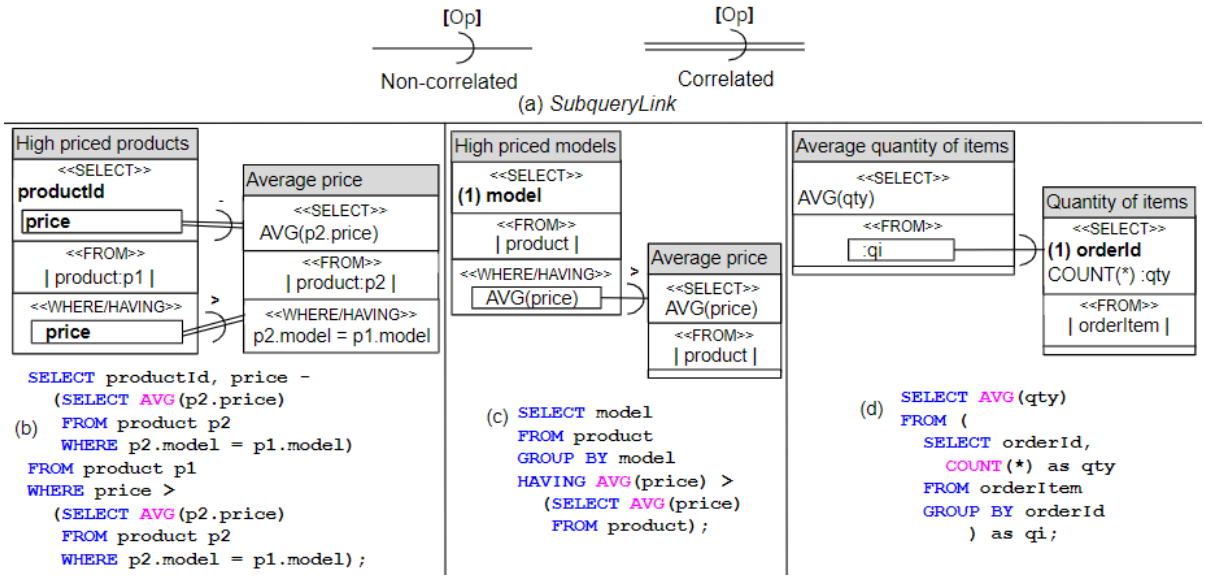
Figura 57 – Construtores para expressões condicionais



Como uma expressão condicional também pode ter muitas operações, sua notação gráfica também é baseada em caixas, compartimentos e setas para modularizar e indicar a sequencias de execução das suas operações. São necessários quatro construtores para especificar expressões condicionais: (1) ConditionalConnective (Figura 57-a) cujo rótulo é igual ao rótulo do construtor LogicalConnective; (2) When (Figura 57-b), que possui três compartimentos: o primeiro corresponde ao seu nome ("WHEN"), o segundo é usado para expressar uma expres-

são lógica a ser avaliada e o terceiro é usado para especificar um construtor DSQL (e.g., um campo, uma subconsulta, uma expressão livre ou outra expressão condicional) a ser executado quando a expressão lógica for verdadeira; (3) Else (Figura 57-c), que possui dois compartimentos: o primeiro corresponde ao seu nome ("ELSE") e o segundo é usado para especificar um construtor DSQL que será executado quando nenhuma das expressões lógicas descritas nos construtores When forem verdadeiras; e (4) CaseLink (Figura 57-d) que mostra a sequência de execução da expressão condicional. O lado direito da Figura 57 ilustra uma consulta DSQL que usa esses construtores e mostra sua respectiva representação em SQL. Essa consulta responde à seguinte pergunta: *"Quais seriam os valores do pedido se as promoções A e B tivessem um desconto de 20% e 10%, respectivamente?"*

Figura 58 – Construtores para subconsultas

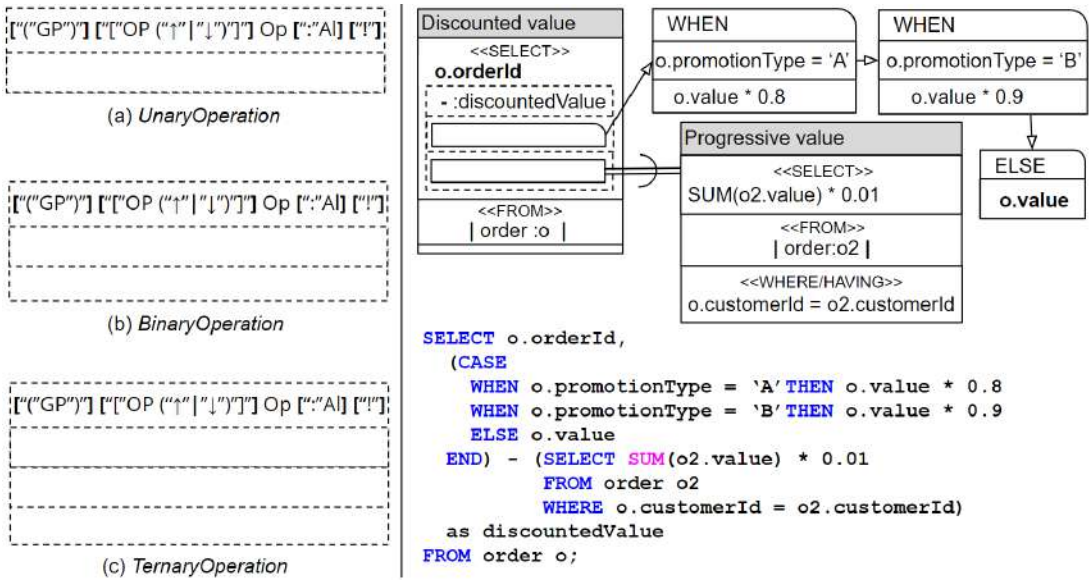


O construtor SubqueryLink (Figura 58-a) conecta os construtores Column ou Source a uma subconsulta mostrando as relações de subconjunto. Quando um SubqueryLink é conectado a um Column, uma operação aritmética ou comparativa pode ser especificada no rótulo ([Op]). A parte inferior da Figura 58 mostra três exemplos de consultas em DSQL que usam o construtor SubqueryLink e suas respectivas representações em SQL. Na Figura 58-b, uma subconsulta correlacionada é reutilizada nos compartimentos Select e Where/Having para representar a seguinte consulta: *"Quais produtos têm preço mais alto que o preço médio para os produtos do mesmo modelo?"*. Na Figura 58-c, uma subconsulta é usada na restrição de agrupamento. Essa figura expressa a seguinte consulta: *"Quais são os modelos de produtos com um preço médio maior que o preço médio geral?"*. Finalmente, na Figura 58-d, uma subconsulta é especificada no compartimento From. Nessa figura, a seguinte consulta é especificada: *"Qual é o número médio de itens por pedido?"*. É importante destacar que uma coluna não pode ser ordenada em uma subconsulta.

Quando todos os operandos de uma operação aritmética, lógica ou comparativa são subconsultas ou expressões condicionais, os construtores UnaryOperation (e.g., *is null*), BinaryO-

peration (e.g., +) ou TernaryOperation (e.g., *between*) devem ser utilizados. O rótulo desses construtores segue a mesma sintaxe do construtor Column e suas notações gráficas são apresentadas nas Figuras 59-a, b e c. O lado direito da Figura 59 mostra uma consulta em DSQL que usa o construtor BinaryOperation e sua representação em SQL. Essa consulta responde à seguinte pergunta: *"Quais seriam os valores dos pedidos se as promoções A e B tivessem um desconto de 20% e 10%, respectivamente, e se um desconto adicional de 1% da soma dos pedidos dos clientes também tivessem sido concedidos?"*.

Figura 59 – Construtores para operações unárias, binárias e ternárias



Uma junção é especificada com o construtor JoinLink (Figura 60-a), que abstrai um tipo de junção (i.e., *Inner join*, *Left outer join*, *Right outer join* ou *Full outer join*), uma manipulação desses tipos (i.e., *Left excluding join*, *Right excluding join* ou *Full excluding join*) ou uma operação semelhante à junção (i.e., *Semijoin*, *Antijoin*, *Division*) (Figura 60-b). Quando é necessário especificar uma *theta* junção ou uma *equi* junção cuja condição não é baseada em chaves primárias e estrangeiras, o construtor ConditionLink (60-c) deve ser usado para especificar essas condições específicas (Figura 60-e). A parte inferior da Figura 60 mostra três exemplos de consultas em DSQL usando esses construtores e suas representações em SQL. Na Figura 60-d, duas junções internas são especificadas para responder à seguinte consulta: *"Qual é o número de itens pedidos agrupados por cliente e pedido?"*. Na Figura 60-e, uma junção interna com subconsulta é especificada para responder à seguinte consulta: *"Para cada cliente, qual é o número médio de itens por pedido?"*. Por fim, na Figura 60-f, um *antijoin* é especificado para responder à seguinte consulta: *"Quais produtos nunca foram vendidos?"*

As operações de conjunto são especificadas com o construtor SetLink (Figura 61-a). Esse construtor permite que duas consultas sejam conectadas e define se a operação especificada (i.e., *Intersect*, *Union*, *Except* ou *Exclusive Union* - Figura 61-b) é distinta (["#"]). A parte inferior da Figura 61 mostra uma consulta DSQL usando o construtor SetLink e sua represen-

tação em SQL. Essa figura ilustra a seguinte consulta: *"Quais são os emails das pessoas que são exclusivamente clientes ou fornecedores?"*

Figura 60 – Construtores para junções

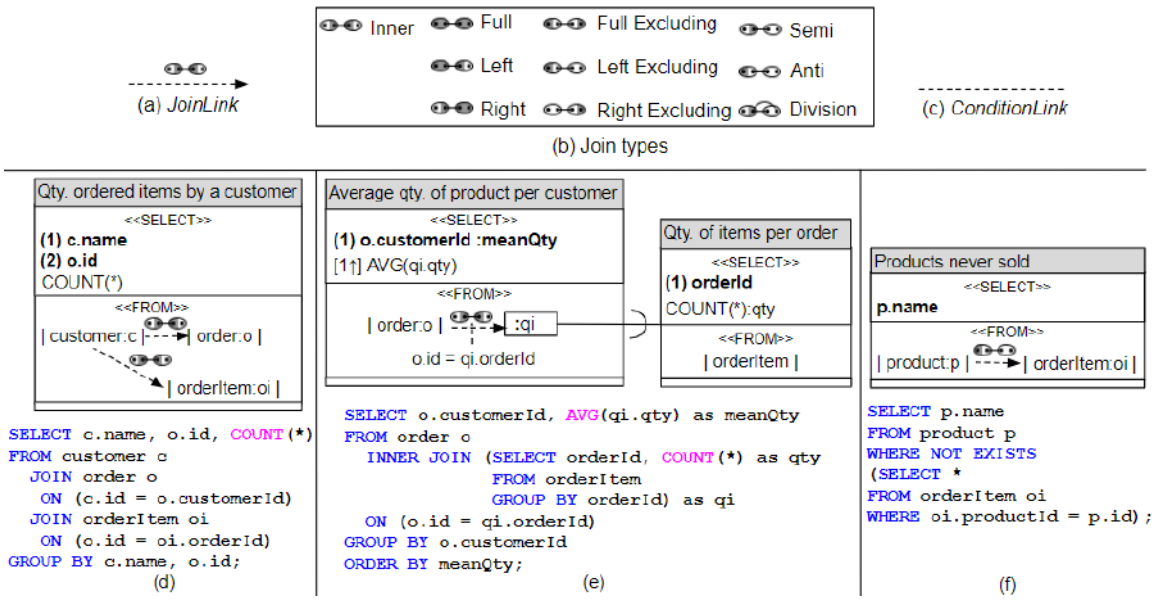
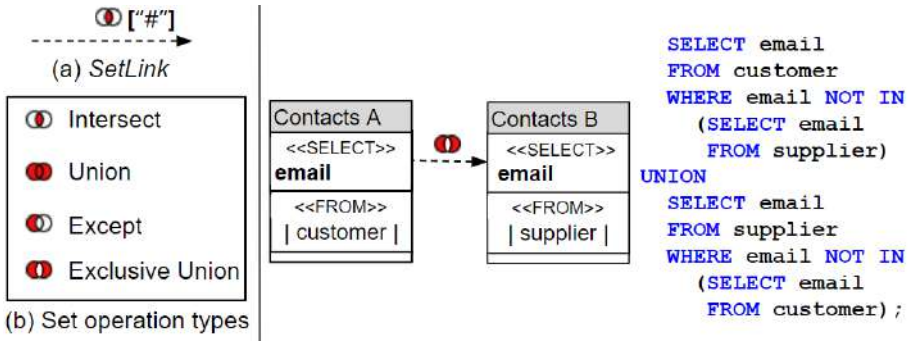


Figura 61 – Construtores para operações de conjunto



4.1.1 Processo de melhoria da sintaxe concreta

O desenvolvimento da sintaxe concreta de DSQL seguiu um processo incremental, no qual foram realizadas seis interações. Desde a primeira interação o objetivo de DSQL foi dar suporte a todos os construtores complexos de SQL usando como base os construtores de caixa, compartimentos e ligações do diagrama de classes de UML. Em cada interação a sintaxe concreta foi refinada para equilibrar a expressividade e a facilidade de uso. Essas mudanças tiveram como base sugestões de especialistas do domínio e alguns dos princípios para construção de notações visuais propostos por Moody (2009) (i.e., claridade semiótica, discriminação perceptiva, transparência semântica e gerenciamento da complexidade).

A claridade semiótica define que deve haver uma correspondência um para um entre os construtores da notação e os conceitos referenciados da linguagem. Desta forma evitou-se

que: 1) um conceito da linguagem possa ser representado por mais de um construtor; 2) dois construtores tenham a mesma representação gráfica; e 3) um conceito da linguagem não tenha construtor correspondente.

A discriminação perceptiva define que construtores diferentes devem ser claramente distinguíveis entre si. Essa distinção pode ser feita com o uso de variáveis visuais (e.g., formas geométricas, cores, rótulos ou uma combinação destes). Seguindo esse princípio, a sintaxe concreta de DSQL foi feita com diferentes formas geométricas e rótulos.

A transparência semântica define que o significado de um construtor pode ser inferido a partir de sua aparência. Esse princípio pode ser seguido com o uso de ícones, setas, ou relacionamentos espaciais. Apesar de DSQL não usar ícones, o uso de setas e relacionamentos espaciais permitem inferir o significado dos construtores após um treinamento inicial.

O gerenciamento da complexidade refere-se à capacidade da sintaxe concreta de representar informações sem sobrecarregar a mente humana. Esse princípio leva em consideração o limite de percepção e o limite cognitivo (WARE, 2004). O limite de percepção refere-se à capacidade de discriminar os elementos do diagrama com relação ao aumento do tamanho do diagrama e o limite cognitivo refere-se ao número de elementos do diagrama que podem ser compreendidos por vez. Seguindo esse princípio, a sintaxe concreta de DSQL permite modularizar a consulta com o uso de compartimentos e ligações.

As mudanças mais significativas que foram realizadas desde a versão inicial são descritas a seguir:

- no construtor *Query*, os compartimentos *Where* e *Having* foram unidos para formar o compartimento *Where/Having*. Essa mudança foi realizada porque as cláusulas *Where* e *Having* têm aplicações distintas. Isto é, a clausula *Having* deve ser usada para realizar seleções sobre funções de agregação, enquanto que as demais condições de seleção são mapeadas para a cláusula *Where* de SQL;

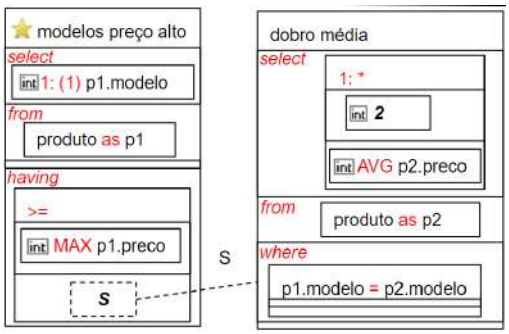
- considerou-se o uso de cores para diferenciar os construtores e pictogramas para diferenciar os tipos do construtores. No entanto, manteve-se uma versão preto e branco com o objetivo de facilitar a especificação usando somente caneta e papel;

- foi preferido o uso de ligações ao invés de embutir caixas. Apesar da notação de caixas ajudar a modularizar diagramas, a abordagem de sucessivamente embutir caixas para representar uma ordem de execução torna a organização dos diagramas mais difícil, principalmente quando se desconhece a profundidade da sequência. Por exemplo, esse problema foi observado nas primeiras versões dos construtores *Case* e *LogicalExpression*, que poderiam ser especificados a partir de caixas embutidas. A solução para esse problema foi o uso de setas, que mostrou-se uma opção melhor para representar uma sequência lógica;

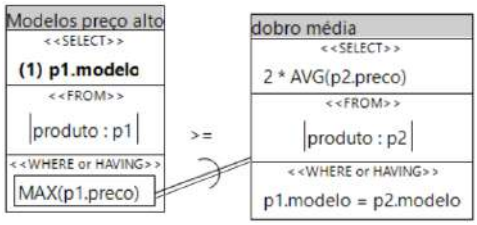
- a especificação de diagramas DSQL foi simplificada com o uso de expressões livres. Apesar de ser possível modelar as condições de seleção e condições de junção usando o construtor *BinaryOperation*, foi observado que é mais rápido e fácil de compreender se estas condições forem escritas com expressões livres. Além disso, o uso de expressões livres aumentou a expressividade de DSQL, uma vez que pode-se especificar qualquer função do banco de dados usando essa alternativa.

Na Figura 62 apresentam-se dois exemplos de consultas em DSQL. Na Figura 62a a consulta é especificada usando a versão 2 de DSQL e na Figura 62b essa mesma consulta é especificada usando a versão atual de DSQL. Nessa consulta pergunta-se: *"quais os modelos de produtos com preço máximo maior que dobro do preço médio dos produtos do mesmo modelo?"*. Na Figura 62c a consulta é especificada usando a versão 4 de DSQL e na Figura 62d essa mesma consulta é especificada usando a versão atual de DSQL. Nessa consulta pergunta-se: *"quais seriam os valores dos pedidos se o tipo de cliente master e vip tivessem um desconto de 20% e 10%, respectivamente?"*.

Figura 62 – Versões da sintaxe concreta de DSQL

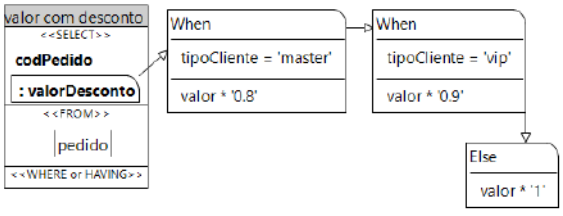


(a) Exemplo 1 na versão 2



(b) Exemplo 1 na versão final

(c) Exemplo 2 na versão 4

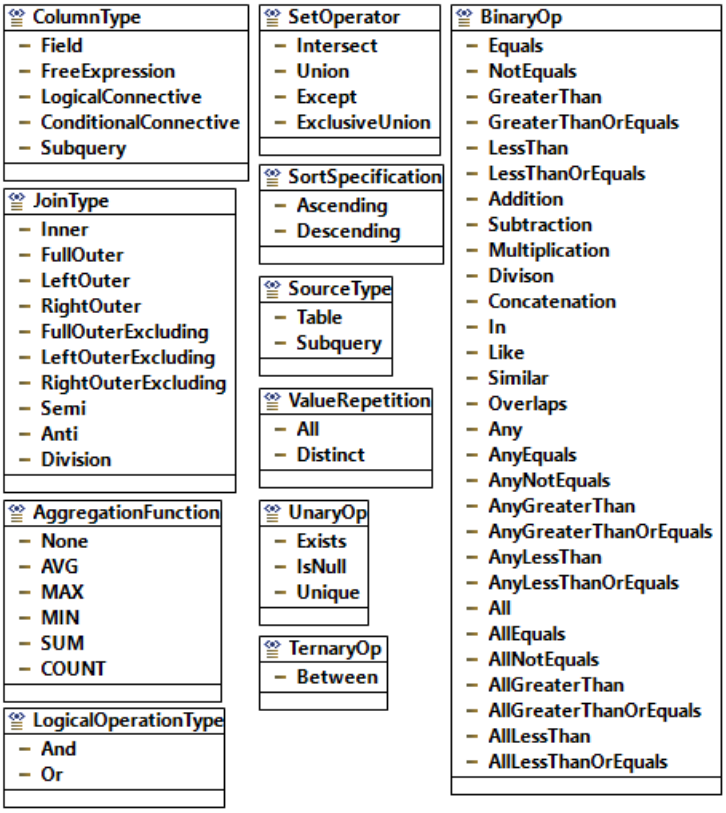


(d) Exemplo 2 na versão final

4.2 SINTAXE ABSTRATA

A sintaxe abstrata de uma linguagem de modelagem descreve os conceitos e as relações válidas entre estes conceitos. Dito isso, nesta seção descreve-se a sintaxe abstrata da linguagem proposta na forma de um metamodelo, que dá suporte a notação especificada na Seção 4.1. Esse metamodelo é formado por 11 enumerações (cf. Figura 63) e 25 metaclasses (cf. Figura 64), sendo 9 metaclasses abstratas (destacadas em cinza) e 16 metaclasses concretas. A explicação do metamodelo é apresentada nos próximos parágrafos. Ressalta-se que metaclasses abstratas não podem ser instanciadas, sendo necessário especializá-las em ao menos uma metaclasse concreta e que todas as associações por composição denotam relacionamentos de contenção (cf. Seção 2.3). Isto é, DSQLDiagram (a área de desenho) contem (direta ou indiretamente) todos os construtores da DSQL, enquanto que as demais associações por composição correspondem aos compartimentos apresentados na sintaxe concreta de DSQL.

Figura 63 – Enumerações do metamodelo de DSQL



Na Figura 63 apresentam-se as enumerações UnaryOp, BinaryOp, TernaryOp, ColumnType, JoinType, AggregationFunction, LogicalOperationType, SetOperator, SortSpecification, SourceType e ValueRepetition. UnaryOp, BinaryOp e TernaryOp representam os tipos de operadores com um, dois ou três operandos, respectivamente, que são permitidos por DSQL. Vale ressaltar que novos operadores (e.g., espaciais ou temporais) podem ser acrescentados em futuras versões da linguagem. ColumnType representa se uma coluna deve ser interpretada como um campo de uma tabela, uma expressão livre, uma expressão lógica, uma expressão

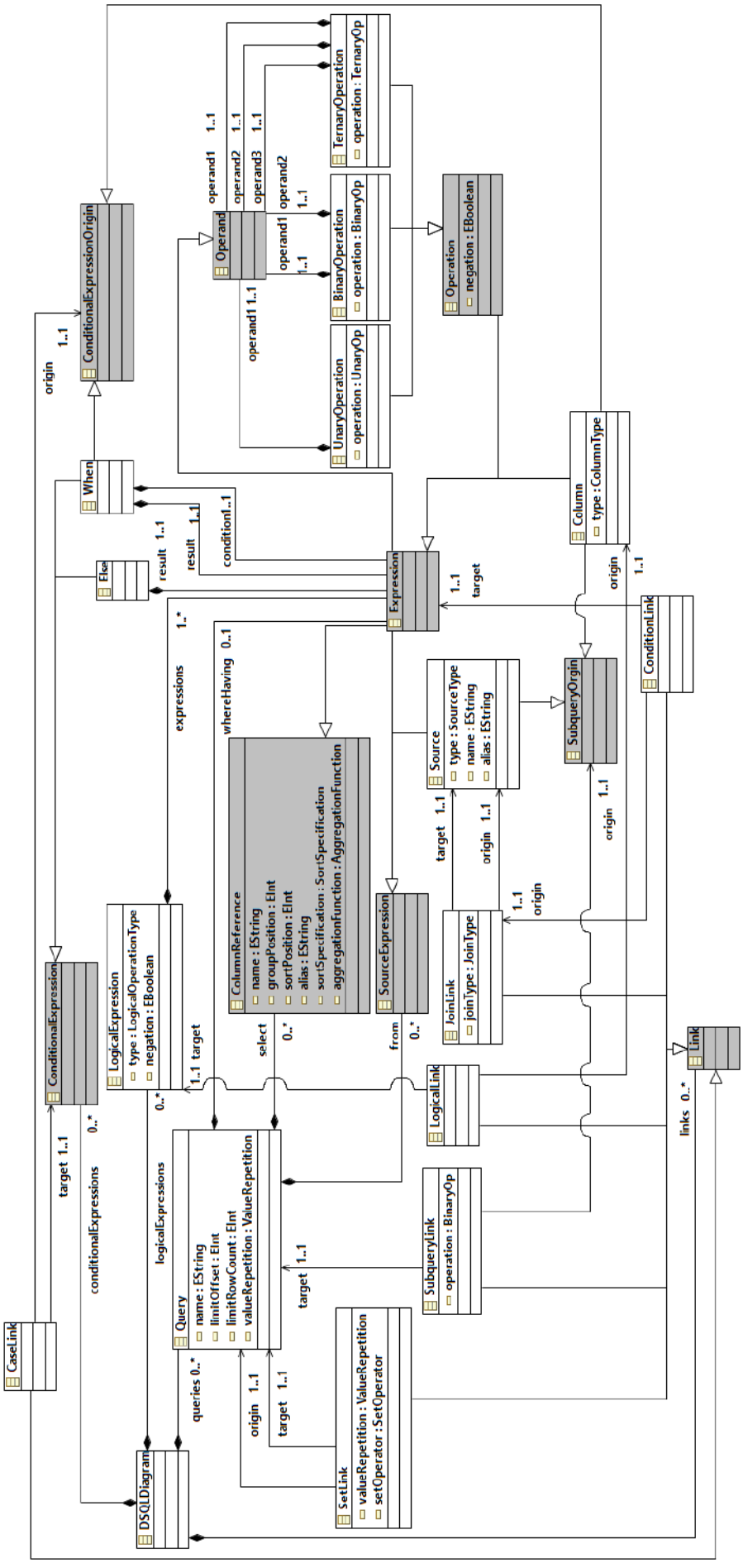
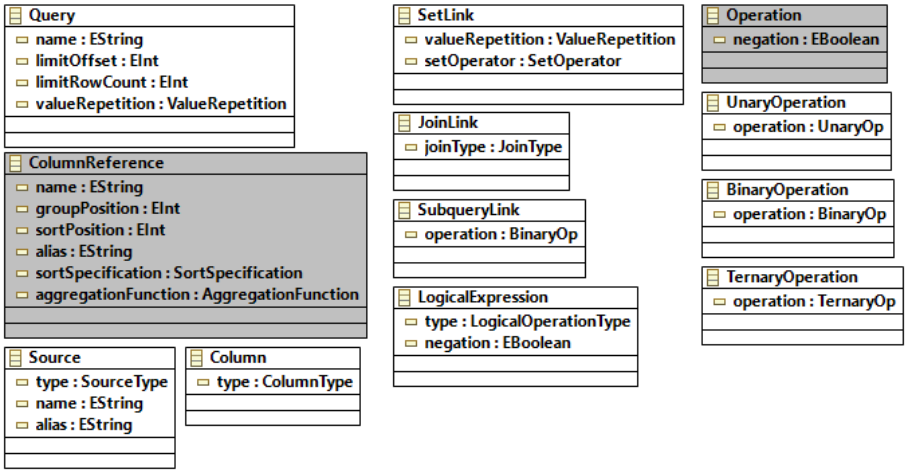


Figura 64 – Metamodelo de DSQL

condicional ou uma subconsulta. JoinType representa os tipos de junção e os tipos de operadores relacionais semelhante a junção permitidos por DSQL. AggregationFuncion representa os tipos de funções de agregação permitidos por DSQL. LogicalOperationType representa se uma expressão lógica é uma conjunção ou uma disjunção. SetOperation representa os tipos de operação de conjunto permitidos por DSQL. SortSpecification representa o tipo de ordenação (i.e., ascendente ou decrescente) de uma coluna na consulta. SourceType representa se uma fonte de dados é uma tabela/visão ou como uma subconsulta. ValueRepetition representa se uma consulta deve retornar valores distintos.

Figura 65 – Atributos do metamodelo de DSQL

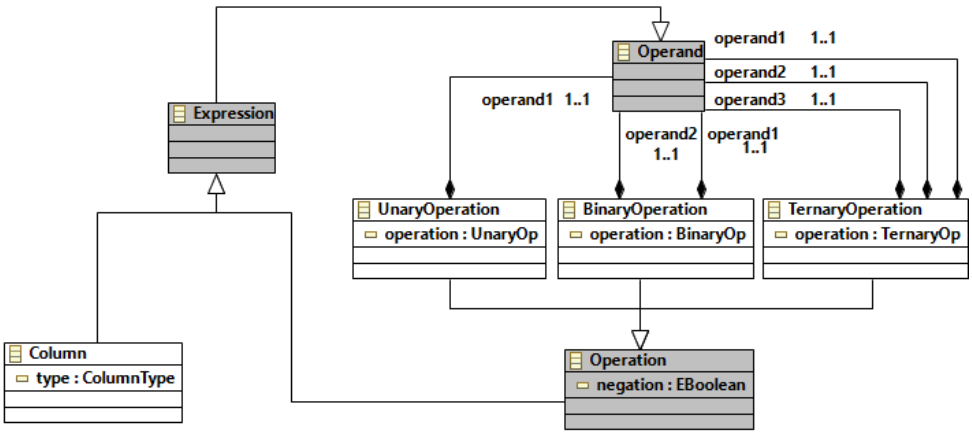


Além dos tipos primitivos, usa-se enumerações para indiciar os tipos de dados simbólicos para os atributos do metamodelo de DSQL. A Figura 65 mostra as classes e seus atributos, os quais são apresentados a seguir.

- o atributo name é do tipo String e é usado para identificar uma consulta (Query), uma fonte de dados (Source) e uma coluna (ColumnReference);
- o atributo alias é do tipo String e é usado para qualificar uma coluna (ColumnReference) e uma fonte de dados (Source);
- os atributos limitOffset e limitRowCount são do tipo Inteiro e são usados para indicar os limites mínimo e máximo da consulta (Query), respectivamente;
- os atributos groupPosition e sortPosition são do tipo Inteiro e são usados para indicar a posição da coluna (ColumnReference) no group by e no order by da consulta. Vale lembrar que a ordem de projeção é dada pela disposição (de cima para baixo) das colunas no compartimento select e que a ordem de projeção pode ser diferente da ordem do group by e do order by;
- o atributo sortSpecification é do tipo simbólico SortSpecification. Este atributo é usado para especificar como cada coluna (ColumnReference) do order by deve ser ordenada, ou seja, ascendente ou decrescente;

- o atributo aggregationFunction é do tipo simbólico AggregationFunction. Este atributo é usado para especificar se uma função de agregação deve ser aplicada na projeção de uma coluna (ColumnReference);
- o atributo joinType é do tipo simbólico JoinType e é usado para especificar o tipo de junção de uma ligação de junção (JoinLink);
- o atributo setOperator é do tipo simbólico SetOperator e é usado para especificar o tipo de operação de conjuntos de uma ligação de conjunto (SetLink);
- o atributo valueRepetition é do tipo simbólico ValueRepetition e é usado para especificar se o resultado de uma consulta (Query) ou de uma operação de conjunto (SetLink) deve retornar todos os valores ou apenas os valores distintos;
- o atributo negation é do tipo Boolean e é usado para negar uma expressão lógica (LogicalExpression) ou operação (Operation);
- o atributo type pode ser dos tipos simbólicos SourceType, ColumnType ou LogicalExpressionType. Quando é um SourceType, este é usado para especificar se uma fonte de dados (Source) é uma tabela[1] ou uma subconsulta. Quando é um ColumnType, este é usado para especificar se uma coluna (Column) é um campo, expressão livre, conectivo lógico, expressão condicional ou subconsulta. Por fim, quando é um LogicalExpressionType, este é usado para especificar se uma expressão lógica (LogicalExpression) é uma conjunção ou disjunção;
- o atributo operation pode ser dos tipos simbólicos UnaryOp, BinaryOp ou TernaryOp, os quais são usados para especificar operações unárias (UnaryOperation), binárias (BinaryOperation) ou ternárias (TernaryOperation). Vale destacar que pode-se usar uma operação binária com subconsultas (SubqueryLink).

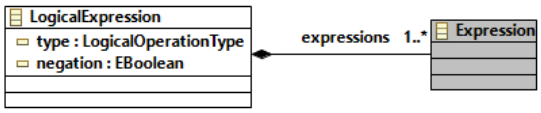
Figura 66 – Metaclasses básicas de uma expressão



[1] para fins práticos DSQL não faz distinção entre Tabela ou Visão.

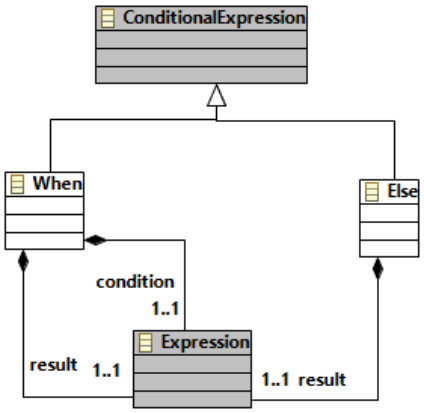
A diagramação de uma expressão em DSQL é uma atividade básica para especificar uma consulta. Na Figura 66 apresenta-se um recorte do metamodelo de DSQL que destaca as metaclasses envolvidas nesta atividade. No metamodelo de DSQL, qualquer expressão é representada pela metaclasse Expression, a qual é uma generalização das metaclasses Column e Operation que, respectivamente, representam uma coluna e um dos três tipos de operações permitidos em DSQL. Os três tipos de operações de DSQL são definidos pela quantidade de operandos (metaclasse Operand). Assim, DSQL permite operações unárias, binárias e ternárias, as quais são respectivamente definidas pelas metaclasses UnaryOperation, BinaryOperation e TernaryOperation, especializações de Operation. Note que a quantidade de operandos é definida pela quantidade de composições que cada uma dessas três metaclasses tem com a metaclasse Operand. Além disso, note também que a metaclasse Expression é uma especialização das metaclasses Operand e ColumnReference, ou seja, uma expressão tanto pode ser diagramada como um operando de uma outra operação quanto uma coluna. Isto é, um campo, uma expressão livre, uma expressão lógica, uma expressão condicional ou uma subconsulta podem ser operandos de qualquer operação. A seguir são apresentadas as metaclasses envolvidas na diagramação de uma expressão lógica ou expressão condicional.

Figura 67 – Metaclasses básicas de uma expressão lógica



Uma expressão lógica em DSQL é especificada a partir da metaclasse LogicalExpression, a qual é composta por ao menos uma expressão (metaclasse Expression), mas admite várias também. Na Figura 67 apresenta-se um recorte do metamodelo de DSQL que destaca essas metaclasses.

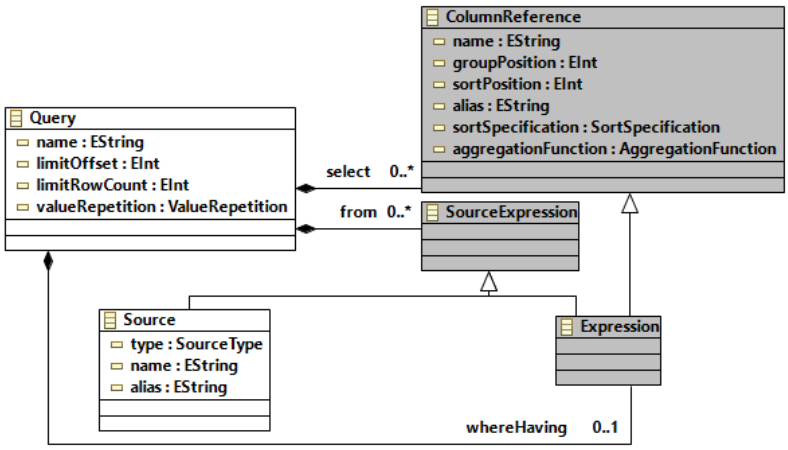
Figura 68 – Metaclasses básicas de uma expressão condicional



Uma expressão condicional em DSQL é representada a partir da metaclasse ConditionalExpression, a qual é uma generalização das metaclasses When e Else. A metaclasse When é composta por dois compartimentos (condition e result) que devem ter apenas uma expressão

cada. Por sua vez, a metaclasse Else é composta por apenas um compartimento (result) que também deve ter apenas uma expressão. Vale ressaltar que, como visto no parágrafo anterior, uma expressão pode ser diagramada como um encadeamento de expressões. Na Figura 68 apresenta-se um recorte do metamodelo de DSQL que destaca essas metaclasses.

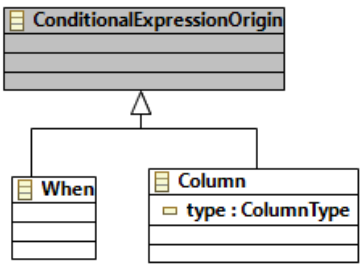
Figura 69 – Metaclasses básicas de uma consulta



Uma consulta em DSQL é especificada a partir da metaclasse Query, a qual é composta por: um compartimento de select que pode conter várias referências de colunas (metaclasse ColumnReference); um compartimento from que pode conter várias especificações de fontes (metaclasse SourceExpression); e um compartimento whereHaving que pode conter uma expressão (metaclasse Expression). Dado que SourceExpression é uma gerenalização das metaclasses Source e Expression, esta permite que sejam especificadas, ao mesmo tempo, no compartimento from, qualquer fonte de dados (i.e., tabela/visão ou subconsulta), bem como expressões que permitem definir condições de junções theta. Na Figura 69 apresenta-se um recorte do metamodelo de DSQL que destaca essas metaclasses.

A metaclasse ConditionalExpressionOrigin é uma generalização das metaclasses When e Column. Este artifício de modelagem visa garantir que a diagramação de um comando CASE nunca tenha como origem um construtor ELSE. Na Figura 70 apresenta-se um recorte do metamodelo de DSQL que destaca essas metaclasses.

Figura 70 – Metaclasse ConditionalExpressionOrigin



A metaclasse SubqueryOrigin é uma generalização das metaclasses Source e Column. Este artificio de modelagem visa garantir que uma subconsulta não seja diagramada apenas como

uma fonte de dados no compartimento from. Isto é, uma subconsuta também pode ser diagramada como uma coluna nos compartimentos select e where/having. Na Figura 71 apresenta-se um recorte do metamodelo de DSQL que destaca essas metaclasses.

Figura 71 – Metaclasse SubqueryOrigin

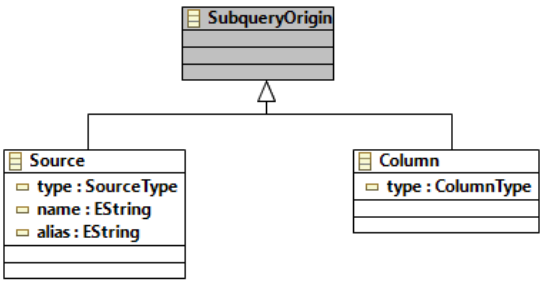
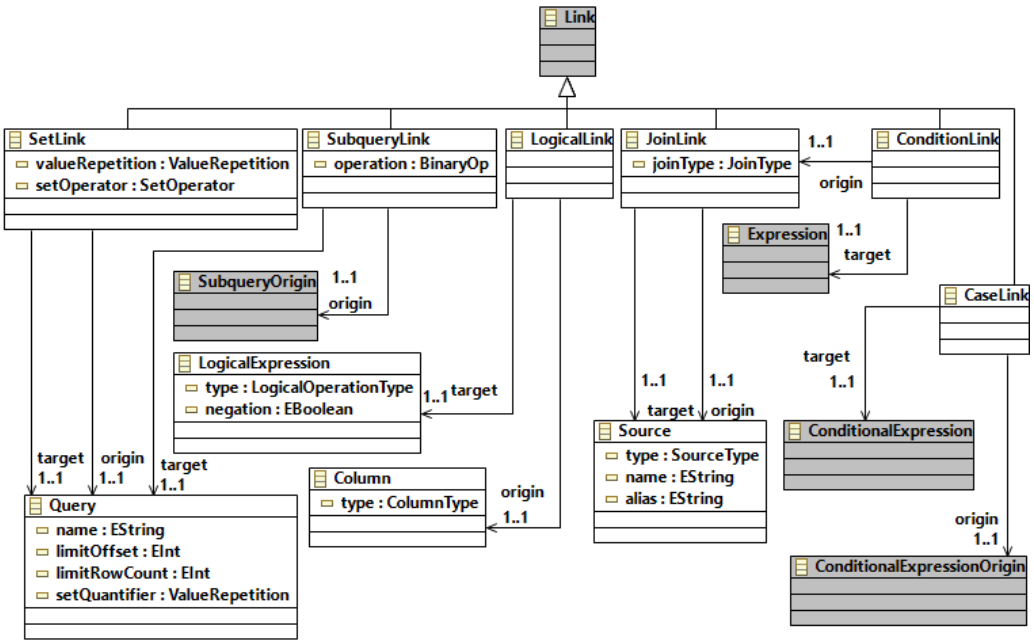


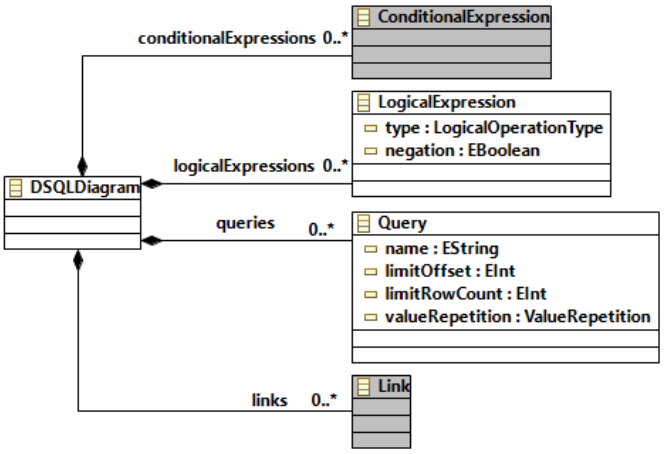
Figura 72 – Metaclasses básicas de uma ligação



Os construtores de DSQL podem estar ligados entre si de várias formas, as quais são diagramadas a partir de uma das seis especializações da metaclasse Link: Setlink, SubqueryLink, JoinLink, ConditionLink, LogicalLink e CaseLink. De modo geral, estas metaclasses ligam um construtor origem (origin) a um construtor alvo (target) e seus nomes sugerem em qual construção devem ser usadas. Neste sentido, uma ligação do tipo conjunto (SetLink) tem como origem e alvo uma consulta (Query). Uma ligação do tipo subconsulta (SubqueryLink) é usada para ligar uma consulta (Query) com uma fonte (Source) ou com uma coluna (Column), pois uma subconsulta é uma consulta (associação target entre SubqueryLink e Query) e Source e Column são especializações de SubqueryOrigin (associação origin entre SubqueryLink e SubqueryOrigin). Uma ligação do tipo junção (JoinLink) é usada para ligar duas fontes (Source) por meio de uma equijunção entre as chaves primárias e estrangeiras. Por isso, este tipo de ligação tem como origem e alvo uma fonte. Contudo, para especificar junções theta ou junções

que não sejam baseadas nas chaves primárias e estrangeiras, deve-se usar uma ligação do tipo condição (ConditionLink), a qual tem como origem uma ligação do tipo junção (JoinLink) e como alvo uma expressão (Expression), a qual permite especificar qualquer operador de comparação e campo de uma tabela. Para encadear expressões lógicas deve-se usar uma ligação do tipo lógica (LogicalLink), a qual tem como origem uma coluna (Column) e como alvo uma expressão lógica (LogicalExpression). Por fim, para diagramar uma expressão condicional deve-se usar uma ligação do tipo Case (CaseLink), a qual tem como origem um construtor When ou Column (especializações de ConditionalExpressionOrigin) e como alvo um construtor Else ou When. Isto é, uma ligação pode partir de um construtor Column, pode partir e chegar de um construtor When e só pode chegar em um construtor Else. Na Figura 72 apresenta-se um recorte do metamodelo de DSQL que destaca essas metaclasses.

Figura 73 – Metaclasses básicas de um diagrama



A metaclasse DSQLDiagram abstrai a área de desenho dos diagramas de DSQL. Isto é, esta metaclasse pode ser composta por muitas consultas/subconsultas (metaclasse Query), muitas expressões lógicas (metaclasse LogicalExpression), muitas expressões condicionais (metaclasse ConditionalExpression) ou muitas ligações (metaclasse Link). Na Figura 73 apresenta-se um recorte do metamodelo de DSQL que destaca essas metaclasses.

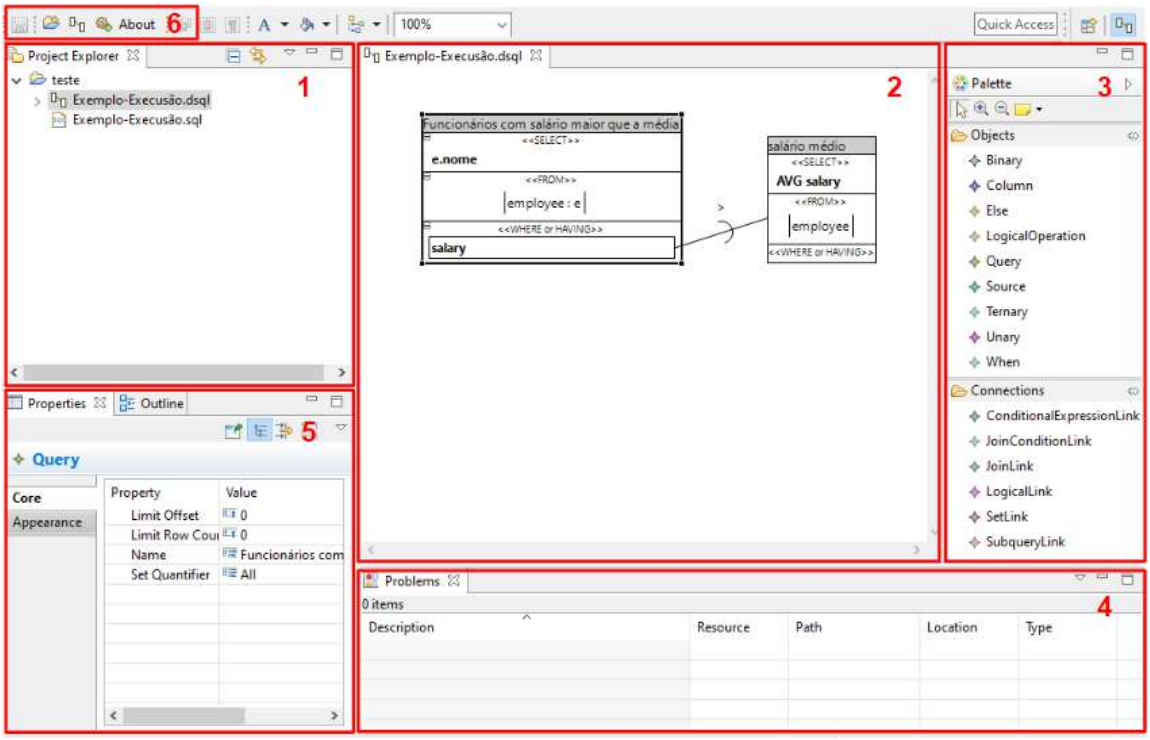
4.3 FERRAMENTA CASE

A DSQL é implementada pela ferramenta DSQLCASE[2]. Essa ferramenta tem como principais funcionalidades: 1) a verificação de erros sintáticos especificados no metamodelo de DSQL; 2) a verificação de erros semânticos ou críticas de boas práticas; 3) a transformação de modelo DSQL em código SQL; e 4) a engenharia reversa de código SQL em modelo DSQL. O desenvolvimento dessa ferramenta orienta-se pelo paradigma MDD (BRAMBILLA; CABOT; WIMMER, 2017), o qual define que uma linguagem de modelagem deve ser especificada a partir de um metamodelo. Nesse sentido, desenvolve-se uma DSML concebida como resultado do processo descrito por Cho, Gray e Syriani (2012).

[2] https://dsqlcase.github.io

O processo de desenvolvimento da DSQLCASE tem início com a codificação do metamodelo de DSQL na linguagem *Emfatic*[3] (cf. Anexo A). O código *Emfatic* corresponde tanto ao metamodelo de DSQL quanto a especificações gráficas da notação. Na sequência, a ferramenta *EuGENia*[4] do *framework* Epsilon é usada para abstrair os detalhes de configuração e execução do GMP. Desta forma, pode-se gerar automaticamente uma versão do editor totalmente funcional a partir de um único metamodelo EMF anotado expresso usando a Emfatic. Além disso, utiliza-se a linguagem EVL para validação dos modelos (cf. Seção 4.3.1) e a biblioteca JSqlParser[5], um parser de SQL para Java, para realizar transformações de DSQL para SQL e vice-versa. Ressalta-se que a leitura e escrita de código SQL é realizada exclusivamente pela biblioteca JSqlParser.

Figura 74 – Ferramenta DSQLCASE - visão geral

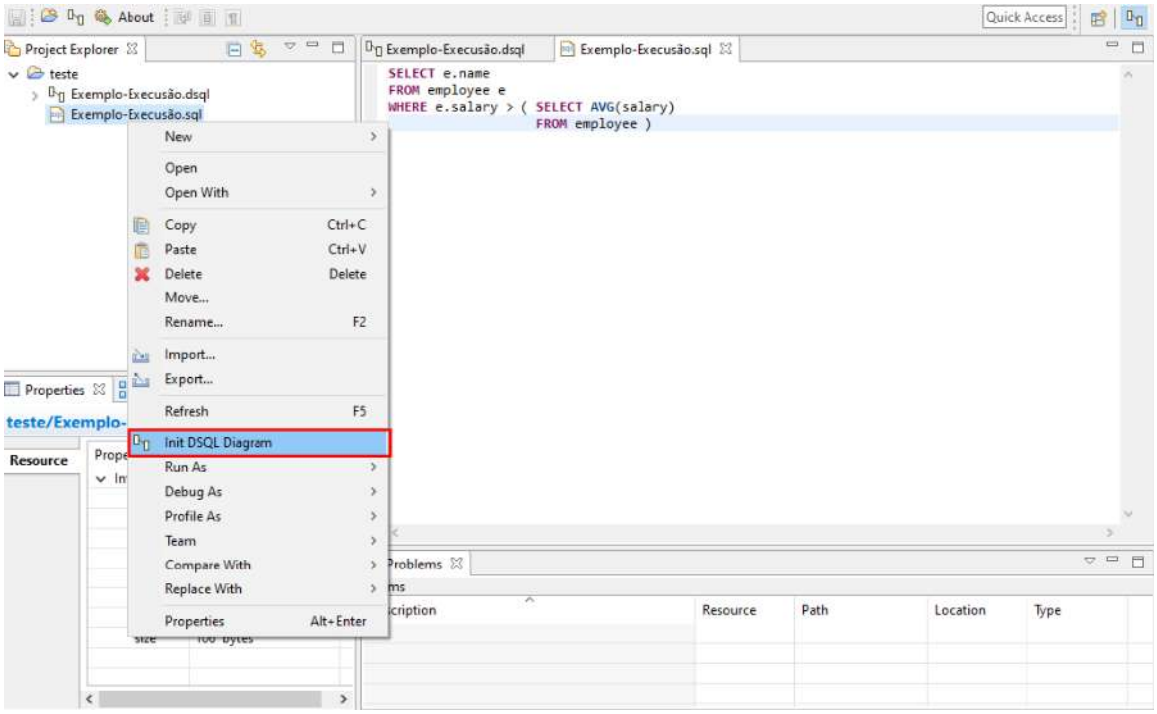


A ferramenta DSQLCASE é formada por seis áreas distintas, onde cada área possui uma utilidade, como pode ser visto na Figura 74. Na área "1" exibem-se os projetos, que mantém os modelos DSQL e códigos SQL. Na área "2", encontra-se a área de desenho, na qual é possível diagramar as consultas em DSQL. Na área "3", mostra-se a paleta que contém os construtores utilizados na modelagem das consultas. Na área "4", apresentam-se os erros semânticos ou os avisos de boas práticas detectados pela ferramenta. Essa área serve como apoio a apresentação dos erros/críticas vistos na Figura 76a. Na área "5", destacam-se as propriedades dos objetos selecionados na área de desenho. Por fim, na área "6", encontram-se os botões *Save*, *Create new project*, *Create DSQL diagram* e *Generate SQL*. Destaca-se que para gerar código SQL

[3] https://www.eclipse.org/emfatic
[4] https://www.eclipse.org/epsilon/doc/eugenia
[5] https://github.com/JSQLParser/JSqlParser

a partir do modelo DSQL selecionado deve-se clicar no botão *Generate SQL* e para gerar o modelo DSQL a partir do código SQL selecionado na área "1" deve-se clicar com o botão direito do *mouse* e escolher a opção *Init DSQL Diagram* (cf. Figura 75).

Figura 75 – Ferramenta DSQLCASE - inicializando diagrama DSQL a partir de SQL



4.3.1 Semântica estática

A semântica estática de uma linguagem de modelagem estabelece as regras que verificam se modelos sintaticamente válidos têm contradições semânticas. Isto é, podem existir modelos que estão de acordo com o metamodelo, mas têm alguma contradição estrutural ou de boa prática. Vale ressaltar que o metamodelo (sintaxe abstrata) define as regras do que pode ser modelado, enquanto que a semântica estática define as regras do que não pode ser modelado. Portanto, a sintaxe abstrata impede a ocorrência de erros, enquanto que a semântica estática informa sobre a ocorrência destes. Nesse contexto, a seguir são apresentadas as regras de boa formação estrutural e de boas práticas consideradas neste trabalho. As regras de boas formação estrutural capturam construções contraditórias que, para manter a simplicidade do metamodelo, não são metamodeladas. Por sua vez, as regras de boas práticas tem como base os trabalhos de Brass e Goldberg (2006) e Paulo (2017). O primeiro trabalho propõe um conjunto de 54 regras, as quais foram catalogadas pelo segundo trabalho. Vale ressaltar que as regras propostas por Brass e Goldberg (2006) não são identificadas de forma exaustiva e que, das regras catalogadas por Paulo (2017), este trabalho considera apenas àquelas que referem-se aos problemas que podem ser identificados de forma textual ou estrutural, pois computacionalmente são menos custosas para serem verificadas, uma vez que não exigem conhecer as particularidades da instância do BD. Portanto, a semântica estática de DSQL

consiste em 21 das 30 regras apresentadas no catálogo de regras definidas por Paulo (2017). Essas regras são listadas a seguir:

- regras de boa formação estrutural:
    1. **Área de desenho vazia**: é necessário especificar ao menos uma consulta (Query) na área de desenho;
    2. **Área de desenho com duas consultas independentes**: não é permitido que exista mais de uma consulta (Query) independente na área de desenho. Isto é, as demais consultas devem ser subconsultas ou alvo de uma ligação de conjunto (SetLink);
    3. **Query sem fonte**: é necessário especificar ao menos uma fonte (Source) na consulta;
    4. **Campo sem nome**: é obrigatório informar o nome do campo quando o construtor Column for do tipo field;
    5. **Tabela sem nome**: é obrigatório informar o nome da tabela quando o construtor Source for do tipo table;
    6. **Subconsulta no compartimento from sem alias**: é obrigatório informar o alias da subconsulta quando o construtor Source for do tipo subquery;
    7. **Column origem de mais de uma CaseLink**: não é permitido que exista uma instância de Column origem de mais de uma ligação de expressão condicional (CaseLink);
    8. **When não associado como alvo de uma CaseLink**: toda instância de When deve ser alvo de uma ligação de expressão condicional (CaseLink);
    9. **When origem de mais de uma CaseLink**: não é permitido que exista uma instância de When origem de mais de uma ligação de expressão condicional (CaseLink);
    10. **When alvo de mais de uma CaseLink**: não é permitido que exista uma instância de When alvo de mais de uma ligação de expressão condicional (CaseLink);
    11. **Else não associado como alvo de uma CaseLink**: toda instância de Else deve ser alvo de uma ligação de expressão condicional (CaseLink);
    12. **Column origem de mais de uma LogicalLink**: não é permitido que exista uma instância de Column origem de mais de uma ligação de expressão lógica (LogicalLink);
    13. **LogicalExpression não associado como alvo de uma LogicalLink**: toda instância de LogicalExpression deve ser alvo de uma ligação de expressão lógica (LogicalLink);

14. **Column origem de mais de uma SubqueryLink**: não é permitido que exista uma instância de Column origem de mais de uma ligação de subconsulta (SubqueryLink);
15. **Source origem de mais de uma SubqueryLink**: não é permitido que exista uma instância de Source origem de mais de uma ligação de subconsulta (SubqueryLink);
16. **SetLink com origem e alvo para mesma Query**: não é permitido que exista uma consulta (Query) origem e alvo de uma mesma ligação de conjunto (SetLink);
17. **JoinLink com origem e alvo para mesma Source**: não é permitido que exista uma fonte (Source) origem e alvo de uma mesma ligação de junção (JoinLink);
18. **Column no compartimento from não associado como alvo de uma ConditionLink**: toda instância de Column no compartimento from deve ser alvo de uma ligação de condição (ConditionLink);

- regras de boas práticas:

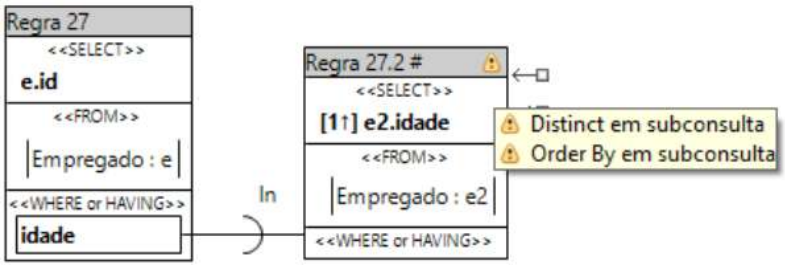
1. **Seleção inconsistente**: aplica-se quando no compartimento Where/Having existem condições mutuamente inconsistente;
2. **Subconsulta não correlacionada com operador Exists**: aplica-se quando uma subconsulta não correlacionada com operador Exists não referencia uma das tabelas (Source) da consulta externa;
3. **Projeção da subconsulta sem campo da subconsulta**: aplica-se quando o compartimento Select da subconsulta não referencia ao menos um campo (Column) da subconsulta;
4. **Projeção sobre um campo que participa de uma seleção de igualdade com um valor constante**: aplica-se quando no compartimento Where/Having um campo (Column) participa de uma igualdade com um valor constante e também aparece no compartimento Select;
5. **Condição de seleção sendo usada em condição de junção**: aplica-se quando é especificada no compartimento From uma condição que deveria estar no compartimento Where/Having;
6. **Projeção com atributos duplicados**: aplica-se quando no compartimento Select um determinado campo (Column) aparece mais de uma vez;
7. **Produto cartesiano desnecessário**: aplica-se quando um produto cartesiano é definido no compartimento From, mas apenas os campos (Column) de uma das tabelas (Source) são usados;
8. **Junção desnecessária**: aplica-se quando, apesar de uma junção ter sido especificada, a projeção e a seleção são feitas considerando apenas os campos (Column) de uma das tabelas (Source);

9. **Condição usando operador Like sem os caracteres curingas**: aplica-se quando o operador Like é utilizado sem os caracteres curingas "%" ou "_";
10. **Subconsulta com operador Exists com projeção diferente de * ou um valor constante**: aplica-se quando a projeção de uma subconsulta com operador Exists contém campos (Column) ou operações;
11. **Subconsulta não correlacionada desnecessária**: aplica-se quando todas as tabelas (Source) da subconsulta estão presentes na consulta externa e a subconsulta não esta correlacionada com a consulta externa;
12. **Funções Avg, Sum e Count com operador Distinct**: aplica-se quando o operador Distinct é utilizado como parte do argumento das funções Avg, Sum e Count. Essa regra visa alertar sobre um possível equívoco, pois apesar de ser semanticamente correto calcular médias, somar e contar valores distintos esta prática é menos frequente;
13. **Subconsultas agrupadas com operador Exists**: aplica-se quando a subconsulta com operador Exists tem um agrupamento;
14. **Agrupamento sem função de agregação**: aplica-se quando a consulta não tem funções de agregação, mas é especificada usando um campo (Column) agrupado;
15. **União desnecessária**: aplica-se quando duas consultas unidas pelo operador Union realizam seleções exclusivas, mas possuem os compartimentos Select e From iguais;
16. **Seleção da consulta externa sendo usada em subconsulta**: aplica-se quando, no compartimento Where de uma subconsulta, especifica-se uma condição que é exclusiva da consulta externa;
17. **Seleção com caracteres especiais, mas sem o uso do operador Like**: aplica-se quando, no compartimento Where, utilizam-se caracteres especiais como "_" ou "%", mas no lugar do operador Like faz-se uma comparação por igualdade. Ressalta-se que é possível ter strings com esses caracteres (e.g., login e senha). Contudo, dado que esta prática não é frequente, essa regra visa alertar sobre um possível equívoco;
18. **Junção externa em contradição com a seleção**: aplica-se quando: 1) faz-se uma junção externa à esquerda e tem-se uma condição de seleção envolvendo apenas a tabela da direita; 2) faz-se uma junção externa à direita e tem-se uma condição de seleção envolvendo apenas a tabela da esquerda; ou 3) faz-se uma junção completa e tem-se duas condições de seleção, uma envolvendo apenas a tabela da esquerda e a outra envolvendo apenas a tabela da direita;
19. **Sempre utilizar alias**: aplica-se quando uma tabela (Source) é usada no compartimento From sem especificar seu alias. Ressalta-se que esta regra visa melhorar a legibilidade e evitar ambiguidades nas consultas;

20. **Subconsulta correlacionada com operador IN**: aplica-se quando, ao invés de especificar a subconsulta correlacionada usando o operador Exist utiliza-se o operador In;
21. **Operador Distinct ou campo ordenado em subconsultas**: aplica-se quando uma subconsulta é especificada usando operador Distinct ou um campo (Column) ordenado foi definido no compartimento Select da subconsulta.

Na Figura 76 ilustra-se uma consulta com deslize de boa prática e a especificação da regra de verificação que permite capturar este deslize. Na Figura 76a tem-se o seguinte deslize de boa prática: operador Distinct ou campo ordenado em subconsulta (i.e. regra 21). Como pode ser observado, a ocorrência do problema é identificada por um ícone amarelo no construtor com problemas, bem como por uma mensagem explicando o motivo da ocorrência do problema. Na Figura 76b mostra-se um fragmento de código em EVL (KOLOVOS et al., 2017), uma implementação de OCL que especifica a regra de verificação em questão. Essa especificação tem início com a definição de um contexto, o qual corresponde a um construtor da linguagem de modelagem (i.e., a uma metaclasse), no caso, o construtor Query (cf. linha 1). Em seguida, verifica-se se a instância do construtor Query é referenciada pela propriedade *source* de alguma instância do construtor *SubquryLink* (i.e., é uma subconsulta) e se a propriedade *valueRepetition* da instância do construtor Query tem valor *Distict* (cf. linhas 4-5). Caso a verificação seja verdadeira, mostra-se uma mensagem explicando o motivo do problema (i.e., "Distinct em subconsulta") (cf. linha 10). De forma semelhante, verifica-se se a instância do construtor Query é referenciada pela propriedade *source* de alguma instância do construtor *SubquryLink* (i.e., é uma subconsulta) e se alguma instância da propriedade *select* da instância do construtor Query tem a propriedade *sortPosition* diferente de 0 (cf. linhas 14-16). Caso a verificação seja verdadeira, mostra-se uma mensagem explicando o motivo do problema (i.e., "Order By em subconsulta") (cf. linha 23).

Figura 76 – Exemplo de deslize de boa prática: *Distinct* ou *Order By* em subconsulta



(a) Apresentação do erro/crítica

```
1  context Query {
2    critique SubQueryDistinct{
3      check{
4        if(SubqueryLink.allInstances().exists(x | x.source = self)
5           and self.valueRepetition = ValueRepetition#Distinct ) {
6          return false;
7        }
8        return true;
9      }
10     message : 'Distinct em subconsulta'
11   }
12   critique SubQueryOrderBy{
13     check{
14       if(SubqueryLink.allInstances().exists(x | x.source = self)) {
15         for(cr in self.select ){
16           if(cr.sortPositition <> 0){
17             return false;
18           }
19         }
20       }
21       return true;
22     }
23     message : 'Order By em subconsulta'
24   }
25 }
```

(b) Especificação da regra em EVL

4.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Nesse capítulo apresentou-se a proposta desse trabalho, uma VQL para especificação de consultas SQL complexas desenvolvida com base no paradigma MDD.

A sintaxe concreta (i.e., notação) foi definida com 15 construtores: Query, Source, Column, LogicalExpression, When, Else, UnaryOperation, BinaryOperation, TernaryOperation, SetLink, SubqueryLink, LogicalLink, JoinLink, ConditionLink e CaseLink (cf. Seção 4.1). Com esses construtores é possível especificar as construções complexas abordadas na Seção 2.1 (i.e., subconsulta, junção, operação de conjunto, expressão condicional e restrição de agrupamento), com exceção de recursão, incluindo a especificação de subconsultas nas cláusulas Select, From, Where e Having e expressão condicional nas cláusulas Select, Where e Having.

No decorrer do desenvolvimento, a sintaxe concreta foi melhorada com base em sugestões de especialistas do domínio e alguns dos princípios para construção de notação visuais proposta por Moody (2009) (i.e., claridade semiótica, discriminação perceptiva, transparência semântica e gerenciamento da complexidade). Desta forma: 1) em relação à claridade semiótica, evitou-se que um conceito da linguagem possa ser representado por mais de um construtor; 2) em

relação à discriminação perceptiva, a notação foi feita com diferentes formas geométricas e rótulos; 3) em relação à transparência semântica, usa-se setas e relacionamentos espaciais; e 4) em relação ao gerenciamento da complexidade, a notação permite modularizar a consulta com uso de compartimentos e ligações.

A sintaxe abstrata (i.e., metamodelo) é formada por 11 enumerações, 9 metaclasses abstratas e 16 metaclasses concretas (cf. Seção 4.2). Consultas especificadas com base nesse metamodelo são, por definição, sintaticamente corretas.

A ferramenta DSQLCASE dá suporte a sintaxe concreta e sintaxe abstrata de DSQL. Além disso, a ferramenta implementa um conjunto inicial de regras de boa formação estrutural (i.e., semântica estática) e dá suporte a geração de código SQL a partir de diagramas DSQL. Vale ressaltar que a semântica estática é formada por 18 regras de boa formação estrutural e de 21 das 30 regras de boa formação catalogadas por Paulo (2017). Destaca-se que a versão atual da DSQLCASE não acessa o banco de dados e, consequentemente, não possui mecanismos para facilitar a especificação ou validação das consultas em DSQL. Essa limitação influencia a usabilidade da ferramenta e, portanto, é uma das funcionalidades a serem adicionadas no futuro.

No próximo capítulo será apresentada a avaliação dessa tese, que consiste em uma análise da expressividade de DSQL com relação aos trabalhos relacionados (cf. Capitulo 3) e dois experimentos para comparar DSQL e SQL com relação ao tempo, acerto e esforço para compreender consultas complexas.

5 AVALIAÇÃO

Neste capítulo apresenta-se a avaliação dessa tese. As seções deste capítulo estão organizadas da seguinte maneira: inicialmente, mostra-se uma análise comparativa da expressividade de DSQL em relação às 21 linguagens selecionadas na RSL (cf. Capítulo 3); na sequência, faz-se dois experimentos para comparar a compreensão de consultas em DSQL e em SQL. O primeiro em relação ao tempo e precisão e o segundo em relação ao tempo, precisão e esforço. Por fim, apresentam-se as considerações finais deste capítulo.

5.1 ANÁLISE COMPARATIVA

Na Tabela 9 apresenta-se uma análise comparativa da expressividade de DSQL em relação aos 21 estudos encontrados na RSL (cf. Capítulo 3). Essa análise mostra que:

- todas as linguagens suportam subconsulta não correlacionada;
- com exceção da linguagem A17, todas as linguagens suportam junção interna;
- com exceção da linguagem A14, todas as linguagens suportam subconsulta na cláusula Where;
- somente as linguagens A01, A02, A03, A04, A07, A08, A11, A12, A15, A19, A21 e DSQL suportam subconsulta correlacionada;
- somente as linguagens A10, A14 e DSQL suportam junção cruzada;
- somente as linguagens A01, A14, A15, A21 e DSQL suportam junção externa;
- somente os linguagens A06, A10, A14, A15, A17, A18, A19 e DSQL suportam operação de conjunto;
- somente os linguagens A01, A03, A06, A07, A12, A15, A20 e DSQL suportam restrição de agrupamento;
- somente a linguagem A03 suporta recursão;
- somente as linguagens A14 e DSQL suportam subconsulta na cláusula Select;
- somente as linguagens A15 e DSQL suportam subconsulta na cláusula Having;
- somente DSQL suporta expressão condicional, subconsulta na cláusula From e expressão condicional nas cláusulas Select, Where e Having; e
- nenhuma linguagem suporta subconsulta ou expressão condicional na cláusula Order By.

Como pode ser observado, embora nenhuma linguagem cubra todos os critérios avaliados, DSQL é a linguagem mais expressiva porque suporta pelo menos 40% mais critérios do que suas principais concorrentes. Ressalta-se que DSQL não suporta subconsulta ou expressão condicional na cláusula Order By e recursão porque a sintaxe do construtor Column não suporta essas construções. Portanto, foi preferido uma notação mais simples do que apoiar essas construções porque elas raramente ocorrem.

#	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	%
A01	●	●	○	●	●	○	●	○	○	○	○	●	○	○	○	○	○	○	33,33
A02	●	●	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	22,22
A03	●	●	○	●	○	○	●	○	●	○	○	●	○	○	○	○	○	○	33,33
A04	●	●	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	22,22
A05	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	16,66
A06	●	○	○	●	○	●	●	○	○	○	○	●	○	○	○	○	○	○	27,77
A07	●	●	○	●	○	○	●	○	○	○	○	●	○	○	○	○	○	○	27,77
A08	●	●	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	22,22
A09	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	16,66
A10	●	○	●	●	○	●	○	○	○	○	○	●	○	○	○	○	○	○	27,77
A11	●	●	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	22,22
A12	●	●	○	●	○	○	●	○	○	○	○	●	○	○	○	○	○	○	27,77
A13	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	16,66
A14	●	○	●	●	●	●	○	○	○	●	○	○	○	○	○	○	○	○	33,33
A15	●	●	○	●	●	●	●	○	○	○	○	●	○	●	○	○	○	○	44,44
A16	●	○	○	●	○	○	○	○	○	○	○	●	○	○	○	○	○	○	16,66
A17	●	○	○	○	○	●	○	○	○	○	○	●	○	○	○	○	○	○	16,66
A18	●	○	○	●	○	●	○	○	○	○	○	●	○	○	○	○	○	○	22,22
A19	●	●	○	●	○	●	○	○	○	○	○	●	○	○	○	○	○	○	27,77
A20	●	○	○	●	○	○	●	○	○	○	○	●	○	○	○	○	○	○	22,22
A21	●	●	○	●	●	○	○	○	○	○	○	●	○	○	○	○	○	○	27,77
DSQL	●	●	●	●	●	●	●	●	○	●	●	●	○	●	●	●	○	●	83,33

○ - evidência não encontrada; ● - evidência encontrada.

C1 - subconsulta não correlacionada; C2 - subconsulta correlacionada; C3 - junção cruzada;
C4 - junção interna; C5 - junção externa; C6 - operação de conjunto;
C7 - restrição de agrupamento; C8 - expressão condicional; C9 - recursão;
C10 - subconsulta na Select; C11 - subconsulta na From; C12 - subconsulta na Where;
C13 - subconsulta na Order By; C14 - subconsulta na Having;
C15 - expressão condicional na Select; C16 - expressão condicional na Where;
C17 -expressão condicional na Order By; C18 - expressão condicional na Having;

Tabela 9 – Análise comparativa da expressividade

5.2 EXPERIMENTO 1

O objetivo do experimento 1 é analisar se a compreensão de consultas em DSQL tem o potencial de ser tão eficaz quanto em SQL. Esse experimento concentra-se em SQL porque

as propostas relacionadas apresentam desvantagens quando comparadas com DSQL (cf. Seção 5.1) e DSQL é tão expressiva quanto SQL (cf. Seção 2.1). Nesse contexto, têm-se as hipóteses nula (H0) e alternativa (H1):

H0: *A compreensão de consultas em DSQL não tem o potencial de ser tão eficaz quanto em SQL.*

H1: *A compreensão de consultas em DSQL tem o potencial de ser tão eficaz quanto em SQL.*

O experimento 1 foi realizado com 57 estudantes de graduação da disciplina de banco de dados, no qual 50 estudantes tinham algum conhecimento de SQL, 7 estudantes não tinham conhecimento de SQL e ninguém conhecia DSQL. Para evitar viés, o planejamento do experimento foi construído usando comparação emparelhada randomizada com dois tratamentos em um fator (WOHLIN et al., 2012), sendo os participantes selecionados aleatoriamente para as equipes A ou B. Antes de iniciar o experimento, cada participante assinou um termo de consentimento (cf. Anexo B) e recebeu um curso de SQL + DSQL de 6 horas (cf. material do curso no Anexo C). Nesse curso, apresentou-se a linguagem DSQL como uma ferramenta auxiliar para ajudar a entender os conceitos de SQL. Durante o experimento, cada participante analisou seis consultas complexas (três em SQL + três em DSQL) (cf. Anexo D) e escolheu uma das quatro alternativas possíveis. Foram registrados a resposta e o tempo de cada questão. Além disso, nenhum *feedback* de resposta e nenhuma ajuda foi dada aos participantes, pois isso seria uma ameaça de validade para os resultados. Ressalta-se que as equipes A e B responderam as mesmas questões, mudando somente a notação requerida para cada questão.

Consulta	CC	V	D	E	NC
Simples	1	178	2	331	Básica
C1	7	180	3	3939	Baixo
C5	3	180	8	4410	Baixo
C4	7	202	5	7298	Médio
C2	5	251	6	12148	Médio
C3	13	315	14	88678	Alto
C6	13	433	17	103778	Alto

Tabela 10 – Estratificação do nível de complexidade das consultas

Na Tabela 10 apresenta-se a estratificação do nível de complexidade das seis consultas do experimento com relação a uma consulta simples. Para calcular a nível de complexidade (NC) são analisados as construções complexas (CC), o volume (V), a Dificuldade (D) e o Esforço (E) de cada consulta. Para isso, utiliza-se uma adaptação da métrica de Halstead (HALSTEAD et al., 1977), em que: n1 é o número de operadores distintos; n2 é o número de operandos distintos; N1 é o número total de operadores; N2 é o número total de operandos; c é o número de construções complexas distintas; e c é o número total de construções complexas. Neste trabalho são considerados como operandos o número de alias, colunas e tabelas e como

operadores o número de operadores e expressões. O nível de complexidade da consulta é calculada como:

$$ComplexidadeConsulta(E) = \frac{n1}{2} * \frac{N2}{n2} * (N1 + N2) * log2(n1 + n2) * ((c + C) + 1) \quad (5.1)$$

Equipe	Part	Notação	Quest	Sucessos	Proporção Precisão	Média Tempo	D.P. Tempo	K-S Tempo
A	29	SQL	87	52	0.59	315.70	130.43	.608
	28	DSQL	84	45	0.51	241.08	95.02	.429
B	28	SQL	84	34	0.40	279.90	133.43	.231
	29	DSQL	87	35	0.41	250.92	144.15	.225
A+B	57	SQL	171	86	0.50	301.50	132.01	.153
	57	DSQL	171	80	0.46	245.38	118.31	.228

Tabela 11 – Estatísticas descritivas do experimento 1

Na Tabela 11 apresentam-se as estatísticas descritivas para as métricas de precisão e tempo. As colunas *Part*, *Quest* e *Sucessos* contêm o número de participantes, o número de questões que esses participantes responderam por notação e o número de respostas corretas desses participantes, respectivamente. A coluna *Proporção* contém a proporção de respostas corretas (*Sucessos* divididas por *Quest*). As últimas três colunas contêm a média (*Média*), o desvio padrão (*D.P.*) e o *p-value* dos testes de normalidade Kolmogorov-Smirnov (*K-S*). Com base nos testes de normalidade, observa-se normalidade no tempo médio das respostas corretas ($\alpha = 0.05$).

Equipe	Two-proportion z-test			Welch t-test (Tempo)		
	Estatística	DF	Significância	Estatística	DF	Significância
A	0.440	1	.507	3.247	92.441	.001
B	0.001	1	.999	0.867	66.847	.388
A+B	0.292	1	.588	2.890	163.780	.004

Tabela 12 – Pontuações dos testes estatísticos do experimento 1

Na Tabela 12 resume-se as pontuações dos testes estatísticos. Para a métrica de precisão aplica-se o teste estatístico *two-proportion z-test* e para a métrica de tempo aplica-se o teste estatístico *Welch t-test* (KITCHENHAM et al., 2017). Como resultados desses testes tem-se que: 1) não foram encontradas evidências estatísticas de diferenças significativas entre as proporções, incluindo a proporção que corresponde à soma dos sucessos de todas as questões (*estatística=0.292, DF=1, p=0.588*); e 2) foram encontradas evidências estatísticas de diferença significativa entre o tempo médio de entendimento das questões (*estatística=2.8904, DF=163.78, p=0.004*).

Os resultados do experimento 1 sugerem que não houve benefício observável na compreensão porque os participantes foram capazes de realizar suas tarefas usando SQL e DSQL.

Uma razão possível para a equivalência de precisão é porque as notações SQL (declarativa) e DSQL (baseada em UML) são fáceis de entender. No entanto, observa-se benefícios em relação à métrica de tempo. Uma razão possível para isso é porque a notação visual de DSQL provê abstrações gráficas modulares que permitem aos seus usuários se concentrarem mais nos aspectos semânticos do que sintáticos da consulta. Nesse contexto, é importante destacar que 88% dos participantes possuíam algum conhecimento de SQL e 100% dos participantes não tinham conhecimento de DSQL. Mesmo com esse contexto desfavorável para o DSQL, pode-se rejeitar a hipótese nula e aceitar a hipótese alternativa. Ou seja, a compreensão de consultas em DSQL tem o potencial de ser tão eficaz quanto em SQL.

5.3 EXPERIMENTO 2

O experimento 2 é semelhante ao experimento 1, com a diferença que no experimento 2 também avalia-se esforço (carga de trabalho mental) para entender as consultas. A carga de trabalho mental é caracterizada como um reflexo do estresse mental relacionado a uma tarefa realizada, seu ambiente e suas condições operacionais específicas, juntamente com a capacidade do trabalhador de responder a essas demandas (CAIN, 2007). Em geral, a carga de trabalho mental associada a uma tarefa fácil é baixa, enquanto tarefas difíceis produzem uma carga de trabalho mental mais alta. O aumento da carga de trabalho mental durante a execução de uma tarefa pode aumentar a fadiga e facilitar o cometimento de erros (CAIN, 2007; LONGO, 2015). Dito isto, o objetivo do experimento 2 foi analisar se a compreensão de consultas em DSQL tem o potencial de ser tão eficaz e fácil de entender quanto em SQL. Nesse contexto, têm-se as hipóteses nula (H0) e alternativa (H1):

H0: *A compreensão de consultas em DSQL não tem o potencial de ser tão eficaz e fácil de entender quanto em SQL.*

H1: *A compreensão de consultas em DSQL tem o potencial de ser tão eficaz e fácil de entender quanto em SQL.*

O experimento 2 foi realizado com 12 estudantes de graduação da disciplina de banco de dados, no qual 7 estudantes tinham algum conhecimento de SQL, 5 estudantes não tinham conhecimento de SQL e ninguém conhecia DSQL. Para evitar viés, este experimento também é construído usando comparação emparelhada randomizada com dois tratamentos em um fator (WOHLIN et al., 2012), sendo os participantes selecionados aleatoriamente para as equipes A ou B. Antes de iniciar o experimento, cada participante também assinou um termo de consentimento (cf. Anexo B) e recebeu o mesmo curso de SQL + DSQL em 6 horas (cf. material do curso no Anexo C). Durante o experimento, cada participante também analisou seis consultas complexas (três em SQL + três em DSQL) (cf. Anexo E) e escolheu uma das quatro alternativas possíveis. Para cada pergunta registrou-se a resposta e o tempo necessário para responder a pergunta. Depois de responder às três perguntas em cada linguagem, os participantes preencheram um questionário do NASA-TLX para coletar *feedback* sobre suas percepções do esforço da tarefa que haviam acabado de concluir. Além disso, nenhum *feedback*

de resposta e nenhuma ajuda foi dada aos participantes, pois isso poderia ameaçar a validade dos resultados.

O NASA *Task Load Index* (NASA-TLX) (HART; STAVELAND, 1988) é um dos instrumentos mais usados para medir a carga de trabalho mental subjetiva. Essa ferramenta consiste em um procedimento de avaliação multidimensional que fornece uma pontuação geral da carga de trabalho com base em uma média ponderada de avaliações em seis dimensões: demandas de tarefas mentais, físicas e temporais; e esforço, frustração e desempenho percebido. O NASA-TLX é aplicado em duas etapas: pontuação e ponderação. Na fase de pontuação, o participante deve atribuir um valor em uma escala de 0 a 100 para a carga de trabalho em cada uma das seis dimensões. A escala é dividida em intervalos de cinco unidades. Na fase de ponderação, o participante define os fatores de carga relevantes. Os participantes são apresentados a quinze comparações binárias das seis dimensões e devem escolher qual de cada par eles percebem como a maior fonte de carga. Isso torna possível ponderar cada um dos fatores em função do número de vezes que eles aparecem. Cada fator pode receber um peso de zero (não aparece nem uma vez e, portanto, não é considerado relevante) até cinco (o fator foi escolhido sempre e, portanto, é considerado a fonte de carga mais importante). O número de vezes que cada fator é selecionado é contado. A pontuação obtida na primeira fase do método (convertida em uma escala de 100) é então multiplicada pelo valor obtido da fase de ponderação para cada dimensão. Dividir a soma de todos os valores por 15 (o número total de combinações binárias) fornece a média ponderada da carga de trabalho global para a tarefa estudada.

Na Tabela 13 apresentam-se as estatísticas descritivas para as métricas de precisão, tempo e esforço de entendimento. As colunas *Part*, *Quest* e *Sucessos* contêm o número de participantes, o número de questões que esses participantes responderam por notação e o número de respostas corretas desses participantes, respectivamente. A coluna *Proporção* contém a proporção de respostas corretas (*Sucessos* divididas por *Quest*). As últimas seis colunas contêm a média (*Média*), o desvio padrão (*D.P.*) e o *p-value* dos testes de normalidade Shapiro-Wilk (*S-W*). Os três primeiros são relacionados ao tempo médio das respostas corretas e os três últimos ao esforço médio de entendimento. Com base nos testes de normalidade, observa-se normalidade no tempo médio das respostas corretas e no esforço médio de entendimento ($\alpha = 0.05$).

Time	Part	Notação	Quest	Sucessos	Proporção Precisão	Média Tempo	D.P. Tempo	S-W Tempo	Média Esforço	D.P. Esforço	S-W Esforço
A	6	SQL	18	15	0.83	257.94	122.07	.116			
	6	DSQL	18	12	0.67	243.35	86.87	.991			
B	6	SQL	18	8	0.44	249.78	151.43	.124			
	6	DSQL	18	9	0.50	269.73	159.59	.163			
A+B	12	SQL	36	23	0.64	255.10	129.59	.132	52.3	17.6	.529
	12	DSQL	36	21	0.58	254.65	120.48	.317	49.6	20.9	.571

Tabela 13 – Estatísticas descritivas do experimento 2

Na Tabela 14 são resumidas as pontuações dos testes estatísticos. Para a métrica de precisão aplica-se o teste estatístico *two-proportion z-test* e para as métricas de tempo e esforço aplica-se o teste estatístico *Welch t-test* (KITCHENHAM et al., 2017). Como resultados destes testes tem-se que: 1) não foram encontradas evidências estatísticas de diferenças significativas entre as proporções, incluindo a proporção que corresponde à soma dos sucessos de todas as questões (*estatística=0.058, DF = 1, p = 0.809*); e 2) não foram encontradas evidências estatísticas de diferença significativa entre o tempo médio e entre o esforço médio de entendimento das questões (i.e., na pontuação da carga de trabalho do NASA-TLX).

Time	Two-proportion z-test			Welch t-test (Tempo)			Welch t-test (Esforço)		
	Estatística	DF	Significância	Estatística	DF	Significância	Estatística	DF	Significância
A	0.593	1	.441	0.362	24.73	.720			
B	0	1	1	0.264	14.92	.795			
A+B	0.058	1	.809	0.012	41.983	.991	0.3383	21.391	.738

Tabela 14 – Pontuações dos testes estatísticos do experimento 2

As fases do protocolo NASA-TLX (i.e., pontuação e ponderação) permitiram observações interessantes além da diferença entre as pontuações médias percebidas da carga de trabalho. Na Figura 77 mostra-se a pontuação média das dimensões do NASA-TLX na fase de pontuação. Nessa tabela observa-se que a demanda mental é a dimensão com maior pontuação média, seguida de esforço, frustração, desempenho, demanda temporal e, finalmente, demanda física. Além disso, observa-se que DSQL tem pontuações médias mais baixas de carga de trabalho na maioria das dimensões (i.e., demanda física, demanda mental, demanda temporal, esforço e frustração) e SQL tem uma pontuação média menor na carga de trabalho apenas na dimensão desempenho. Na Figura 78 mostra-se a porcentagem da seleção das dimensões do NASA-TLX na fase de ponderação. Nessa tabela observa-se que a demanda mental é a dimensão com a maior porcentagem (22,2%), seguida pelo desempenho (20,6%), frustração (18,3%), esforço (17,2%), demanda física (11,7%) e, finalmente, demanda temporal (10%). A análise das duas fases sugere que a demanda mental é percebida como a dimensão mais exigente e a dimensão mais relevante na carga de trabalho, enquanto que a demanda temporal e a demanda física são percebidas como as dimensões menos exigentes e menos relevantes na carga de trabalho.

Os resultados do experimento 2 sugerem que não houve benefício observável na eficácia e facilidade de entendimento porque os participantes foram capazes de realizar suas tarefas usando SQL e DSQL. Uma razão possível para essa equivalência é porque as notações SQL (declarativa) e DSQL (baseada em UML) são fáceis de entender. De acordo com Eitrheim e Fernandes (2016), os níveis de carga de trabalho abaixo de 50 são percebidos como aceitáveis e tanto SQL quanto DSQL possuem cargas de trabalho próximas a esse nível (i.e., 52,3 e 49,6, respectivamente). Nesse contexto, é importante destacar que 58% dos estudantes possuíam algum conhecimento de SQL e 0% dos estudantes possuíam conhecimento de DSQL. Mesmo com esse contexto desfavorável para DSQL, pode-se rejeitar a hipótese nula e aceitar a hipótese

Figura 77 – Pontuação média das dimensões do NASA-TLX na fase de pontuação

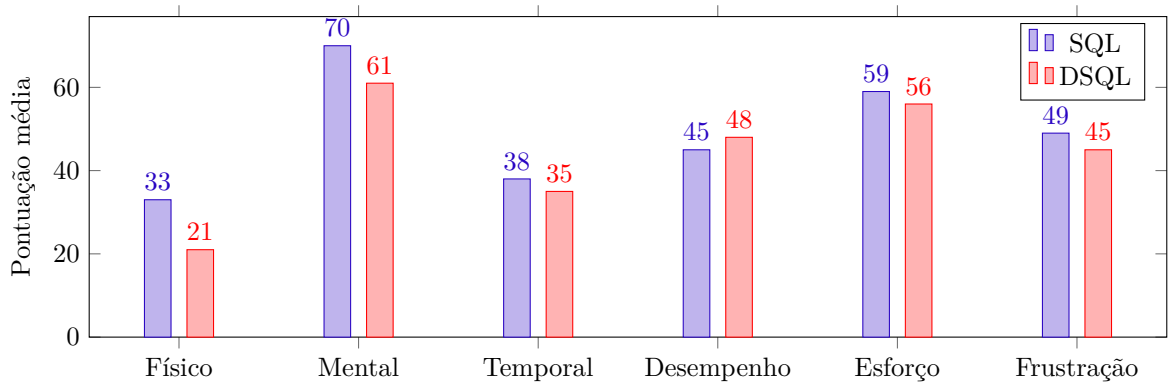
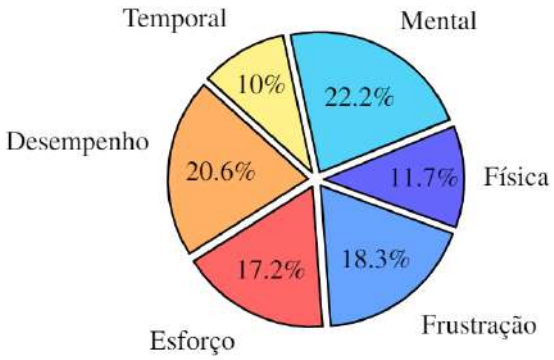


Figura 78 – Porcentagem da seleção das dimensões do NASA-TLX na fase de ponderação



alternativa. Ou seja, DSQL tem o potencial de ser tão eficaz e fácil de entender quanto o SQL.

5.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

No presente capítulo efetuou-se a avaliação de DSQL. Inicialmente fez-se uma análise comparativa de DSQL em relação às 21 linguagens selecionadas na RSL. Como resultados dessa análise têm-se que DSQL é a linguagem mais expressiva porque suporta pelo menos 40% mais critérios do que os concorrentes. Em seguida, realiza-se dois experimentos para comparar a compreensão de consultas em DSQL e em SQL. O primeiro em relação ao tempo e precisão e o segundo em relação ao tempo, precisão e esforço. Como resultados do primeiro experimento têm-se que as duas linguagens são equivalentes com relação a precisão, mas DSQL mostrou-se mais eficaz com relação ao tempo necessário para responder às questões. Como resultados do segundo experimento têm-se que as duas linguagens são equivalentes com relação à precisão, tempo e esforço. Além disso, no segundo experimento foi constatado que a carga de trabalho percebida nas duas linguagens são aceitáveis (i.e., menor que 50).

No capítulo seguinte serão apresentadas as conclusões, bem como as contribuições, as limitações e os trabalhos futuros.

6 CONCLUSÃO

Este capítulo apresenta as considerações finais, as contribuições, as limitações e os trabalhos futuros relacionados a esse trabalho.

6.1 CONSIDERAÇÕES FINAIS

Dada a relevância acadêmica e industrial de SQL e a natureza potencialmente complexa das consultas escritas nessa linguagem, as VQL que favorecem seu uso e, consequentemente, diminuem a carga de trabalho mental são importantes ferramentas pedagógicas e tecnológicas. No entanto, embora várias VQL tenham sido propostas, uma revisão do estado da arte verificou que essas VQL não são amplamente utilizadas na prática, pois não abrangem várias construções complexas e não possuem ferramentas CASE disponíveis para seus usuários finais, comprometendo sua expressividade e disponibilidade. Nesse contexto, nessa tese propôs-se a DSQL, uma VQL que visa ser uma alternativa visual/diagramática para modelar consultas complexas que seja tão compreensível e eficaz quanto SQL sem aumento do esforço.

A notação de DSQL é baseada nas abstrações visuais do diagrama de classes UML: caixas, compartimentos e ligações e seus principais pontos fortes são:

- modularidade por meio de construtores de caixa, o que permite restringir a análise a um aspecto específico da consulta, criando condições para aprimorar as habilidades de raciocínio e solução de problemas antes de codificar a consulta em SQL;
- extensibilidade por meio de construtores de ligações, o que permite especificar facilmente grandes quantidades de conexões entre construtores de caixas, favorecendo a compreensão de relacionamentos de subconjuntos/conjunto, além de definir a sequência lógica de execuções;
- reutilização de construções, que evita redundância, aumenta a produtividade e diminui a carga cognitiva;
- simplicidade por meio de uma notação que pode ser desenhada à mão ou implementada por uma ferramenta CASE;
- expressividade por meio de uma notação que abrange a maioria das construções SQL (pelo menos 40% mais expressiva que suas principais concorrentes).

Em dois experimentos a compreensão de consultas em DSQL foi comparada com SQL. No primeiro foi avaliado o tempo e a precisão e no segundo foi avaliado o tempo, a precisão e o esforço. Como resultados desses experimentos têm-se que não existe diferença significativa na precisão, tempo e esforço, mas DSQL é significativamente mais rápida e fácil de compreender

do que SQL. Ressalta-se que o protocolo NASA-TLX foi usado para medir o carga de trabalho mental percebida pelos participantes (i.e., esforço). De acordo com Eitrheim e Fernandes (2016), a pontuação da carga de trabalho percebida nas duas linguagens é aceitável (i.e., ≤ 50). Além disso, a DSQL possui pontuações médias de carga de trabalho mais baixas do SQL em quase todas as dimensões analisadas do NASA-TLX (i.e., demanda física, demanda mental, demanda temporal, esforço e frustração), com SQL sendo melhor apenas na dimensão desempenho.

Em resumo, a avaliação experimental evidenciou que DSQL tem o potencial de ser tão compreensível e eficaz quanto SQL sem aumentar o esforço e que DSQL avança o estado da arte das VQL, pois diminui as deficiências encontradas nas propostas relacionadas.

6.2 CONTRIBUIÇÕES

De maneira geral, como contribuições desse trabalho têm-se um conjunto de recursos tecnológicos e visuais que permitem facilitar a especificação e a compreensão de consultas SQL complexas. A seguir são listadas as contribuições mais evidentes:

- Uma RSL que investigou a expressividade notacional, o suporte tecnológico e a avaliação científica das VQL para investiga as limitações das VQL usadas para especificar consultas SQL complexas. Ressalta-se que as evidências apresentadas foram guiadas por dois exemplos de execução. Estes exemplos foram usados porque os exemplos apresentados nos artigos selecionados são muito específicos (i.e., não são adequados para realizar análises comparativas imparciais);
- Uma VQL denominada DSQL, para especificação de consultas SQL complexas que supera as deficiências identificadas na RSL;
- Um metamodelo que descreve DSQL de forma inequívoca e serve como base para implementação da ferramenta CASE;
- Uma ferramenta CASE que dá suporte à modelagem de consultas em DSQL. Essa ferramenta implementa o metamodelo proposto e é desenvolvida com base no paradigma MDD. Além disso, implementa um conjunto inicial de regras de boa formação estrutural (i.e., semântica estática) e dá suporte a geração de código SQL a partir de diagramas DSQL.

Publicações

A seguir são listados as publicações relacionadas com essa tese:

- Silva E, Franco N, Ferro M, Fidalgo R. Mental Workload Impact of a Visual Language on Understanding SQL Queries. In Brazilian Symposium on Computers in Education

(Simpósio Brasileiro de Informática na Educação-SBIE) 2019 Nov 11 (Vol. 30, No. 1, p. 239).

Em processo de revisão:

- Silva E, Franco N, Ferro M, Fidalgo R. Specification of Complex SQL Queries using Visual Query Languages: A Systematic Literature Review. In ACM Transactions on Database Systems.

É importante mencionar que apenas os resultados preliminares deste estudo foram publicados nesses artigos. Assim, outros artigos estão em desenvolvimento para publicação das principais contribuições deste trabalho.

6.3 LIMITAÇÕES

Como limitações desse trabalho têm-se que:

- A notação de DSQL não suporta a especificação de recursão, subconsulta na cláusula Order By e expressão condicional na cláusula Order By. Apesar dessa limitação comprometer a expressividade de DSQL, foi preferível manter a simplicidade da notação de DSQL do que dar suporte a esses construtores porque essas construções são raras de ocorrer e complexas de serem representadas;
- A semântica estática de DSQL não foi definida de forma sistemática e, portanto, podem existir regras de boa formação que não foram identificadas. Ressalta-se que a definição sistemática da semântica estática de DSQL está fora do escopo dessa tese;
- As regras de transformação de DSQL para SQL e vice-versa também não foram definidas de forma sistemática. Portanto, apesar de existir uma correspondência 1:1 entre os construtores de DSQL e SQL, podem existir modelos que não sejam transformadas no código SQL esperado e vice-versa. Ressalta-se que a definição sistemática das regras de transformação de DSQL estão fora do escopo dessa tese.

6.4 TRABALHOS FUTUROS

O metamodelo proposto foi construído de forma extensível, uma vez que seu foco está na estrutura base da consulta e não nas operações, portanto, pode-se adicionar novas operações editando somente a enumeração referente ao tipo de operação (i.e., unária, binária ou ternária). Como um estudo futuro pode-se estender o metamodelo para dar suporte a outros domínios (e.g., geográfico, temporal ou NoSQL).

Os experimentos consideraram apenas a compreensão de consultas SQL. Como um estudo futuro pode-se considerar a avaliação da especificação de consultas SQL, bem como as interações entre a compreensão e a especificação.

REFERÊNCIAS

ABOUZIED, A.; HELLERSTEIN, J.; SILBERSCHATZ, A. Dataplay: Interactive tweaking and example-driven correction of graphical database queries. In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. New York, NY, USA: ACM, 2012. (UIST '12), p. 207–218. ISBN 978-1-4503-1580-7. Disponível em: <http://doi.acm.org/10.1145/2380116.2380144>.

ACHILLEOS, A.; GEORGALAS, N.; YANG, K. An open source domain-specific tools framework to support model driven development of oss. In: _____. *Model Driven Architecture-Foundations and Applications: Third European Conference, ECMDA-FA 2007, Haifa, Israel, June 11-15, 2007, Proccedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 1–16. ISBN 978-3-540-72901-3. Disponível em: <https://doi.org/10.1007/978-3-540-72901-3_1>.

ACM/IEEE. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. NY, USA: ACM, 2013. 999133. ISBN 978-1-4503-2309-3.

ACM/IEEE. *Information Technology Curricula 2017: Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology*. NY, USA: ACM, 2017. ISBN 978-1-4503-6416-4.

ARMBRUST, M.; XIN, R. S.; LIAN, C.; HUAI, Y.; LIU, D.; BRADLEY, J. K.; MENG, X.; KAFTAN, T.; FRANKLIN, M. J.; GHODSI, A. et al. Spark sql: Relational data processing in spark. In: ACM. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. [S.l.], 2015. p. 1383–1394.

BACON, D. F.; BALES, N.; BRUNO, N.; COOPER, B. F.; DICKINSON, A.; FIKES, A.; FRASER, C.; GUBAREV, A.; JOSHI, M.; KOGAN, E. et al. Spanner: Becoming a sql system. In: ACM. *Proceedings of the 2017 ACM International Conference on Management of Data*. [S.l.], 2017. p. 331–343.

BADRE, A. N.; CATARCI, T.; MASSARI, A.; SANTUCCI, G. Comparative ease of use of a diagrammatic vs. an iconic query language. In: *IDS*. [S.l.: s.n.], 1996. p. 4.

BAKKE, E.; KARGER, D. R. Expressive query construction through direct manipulation of nested relational results. In: *Proceedings of the 2016 International Conference on Management of Data*. New York, NY, USA: ACM, 2016. (SIGMOD '16), p. 1377–1392. ISBN 978-1-4503-3531-7. Disponível em: <http://doi.acm.org/10.1145/2882903.2915210>.

BLOESCH, A. C.; HALPIN, T. A. Conceptual queries using conquer-ii. In: SPRINGER. *International Conference on Conceptual Modeling*. [S.l.], 1997. p. 113–126.

BORGES, C. R.; MACÍAS, J. A. Feasible database querying using a visual end-user approach. In: *Proceedings of the 2Nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. New York, NY, USA: ACM, 2010. (EICS '10), p. 187–192. ISBN 978-1-4503-0083-4. Disponível em: <http://doi.acm.org/10.1145/1822018.1822047>.

BRAMBILLA, M.; CABOT, J.; WIMMER, M. Model-Driven Software Engineering in Practice: Second Edition. *Synthesis Lectures on Software Engineering*, v. 3, n. 1, p. 1–207, 2017. ISSN 2328-3319. Disponível em: <http://www.morganclaypool.com/doi/10.2200/S00751ED2V01Y201701SWE004>.

BRASS, S.; GOLDBERG, C. Semantic errors in SQL queries: A quite complete list. *Journal of Systems and Software*, v. 79, n. 5, p. 630–644, 2006. ISSN 01641212.

BUDINSKY, F. *Eclipse Modeling Framework: A Developer's Guide*. [S.l.]: Addison-Wesley, 2004. (The eclipse series). ISBN 9780131425422.

CAIN, B. *A review of the mental workload literature*. [S.l.], 2007.

CATARCI, T.; COSTABILE, M. F.; LEVIALDI, S.; BATINI, C. Visual query systems for databases: A survey. *Journal of Visual Languages & Computing*, Elsevier, v. 8, n. 2, p. 215–260, 1997.

CATARCI, T.; SANTUCCI, G. Diagrammatic vs textual query languages: A comparative experiment. In: _____. *Visual Database Systems 3: Visual information management*. Boston, MA: Springer US, 1995. p. 69–83. ISBN 978-0-387-34905-3. Disponível em: <http://dx.doi.org/10.1007/978-0-387-34905-3_5>.

CELKO, J. *Joe Celko's SQL for smarties: advanced SQL programming*. [S.l.]: Elsevier, 2010.

CERULLO, C.; PORTA, M. A system for database visual querying and query visualization: Complementing text and graphics to increase expressiveness. In: *18th International Workshop on Database and Expert Systems Applications (DEXA 2007)*. [S.l.: s.n.], 2007. p. 109–113. ISSN 1529-4188.

CETINKAYA, D.; VERBRAECK, A. Metamodeling and model transformations in modeling and simulation. In: *Proceedings of the 2011 Winter Simulation Conference (WSC)*. [S.l.: s.n.], 2011. p. 3043–3053. ISSN 0891-7736.

CHAMBERLIN, D. D.; BOYCE, R. F. Sequel: A structured english query language. In: *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control*. [S.l.: s.n.], 1974. p. 249–264.

CHO, H.; GRAY, J.; SYRIANI, E. Creating visual domain-specific modeling languages from end-user demonstration. In: *Proceedings of the 4th International Workshop on Modeling in Software Engineering*. Piscataway, NJ, USA: IEEE Press, 2012. (MiSE '12), p. 22–28. ISBN 978-1-4673-1757-3. Disponível em: <http://dl.acm.org/citation.cfm?id=2664431.2664435>.

COCKROACH. *CockroachDB: Distributed SQL*. 2020. Https://www.cockroachlabs.com/product/. Accessed: 2020-01-01.

CZEJDO, B. D.; TUCCI, R. P.; EMBLEY, D. W.; LIDDLE, S. W. Graphical query specification with participation constraints. In: *Computing and Information, 1993. Proceedings ICCI '93., Fifth International Conference on*. [S.l.: s.n.], 1993. p. 433–437.

DANAPARAMITA, J.; GATTERBAUER, W. Queryviz: Helping users understand sql queries and their patterns. In: *Proceedings of the 14th International Conference on Extending Database Technology*. New York, NY, USA: ACM, 2011. (EDBT/ICDT '11), p. 558–561. ISBN 978-1-4503-0528-0. Disponível em: <http://doi.acm.org/10.1145/1951365.1951440>.

DARMAWIKARTA, D. *Oracle SQL, A Beginner's Tutorial*. [S.l.]: Brainy Software Inc, 2016.

DATE, C. J. *SQL and relational theory: how to write accurate SQL code*. 3st. ed. [S.l.]: "O'Reilly Media, Inc.", 2015.

DEAN, J.; GHEMAWAT, S. Mapreduce: Simplified data processing on large clusters. 2004.

EITRHEIM, M. H.; FERNANDES, A. The nasa task load index for rating workload acceptability. In: *Human Factors and User Needs in Transport, Control, and the Workplace*. [S.l.: s.n.], 2016.

EL-MAHGARY, S.; SOISALON-SOININEN, E. A form-based query interface for complex queries. *J. Vis. Lang. Comput.*, Academic Press, Inc., Orlando, FL, USA, v. 29, n. C, p. 15–53, ago. 2015. ISSN 1045-926X. Disponível em: <http://dx.doi.org/10.1016/j.jvlc.2015.03.001>.

ELMASRI, R.; NAVATHE, S. B. *Fundamentals of Database Systems, Seventh Edition*. Boston, MA, USA: Person, 2016. ISBN 0133970779.

EMF. *Eclipse Modeling Framework Project*. 2011. Http://www.eclipse.org/ modeling/emf/. Accessed: 2016-12-14.

EPSTEIN, R. G. The tabletalk query language. *Journal of Visual Languages & Computing*, v. 2, n. 2, p. 115 – 141, 1991. ISSN 1045-926X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1045926X05800266>.

FONTANA, F. A.; BRUNELIERE, H.; MüLLER, H.; RAIBULET, C. Guest editors' introduction to the special issue on model driven engineering and reverse engineering: Research and practice. *Journal of Systems and Software*, v. 159, p. 110446, 2020. ISSN 0164-1212. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0164121219302201>.

GRYZ, J.; WANG, Q.; QIAN, X.; ZUZARTE, C. Sql queries with case expressions. In: *Foundations of Intelligent Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 351–360. ISBN 978-3-540-68123-6.

HALPIN, T.; BLOESCH, A. Data modeling in uml and orm: a comparison. *Journal of Database Management (JDM)*, IGI Global, v. 10, n. 4, p. 4–13, 1999.

HALSTEAD, M. H. et al. *Elements of software science*. [S.l.]: Elsevier New York, 1977. v. 7.

HART, S. G.; STAVELAND, L. E. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In: HANCOCK, P. A.; MESHKATI, N. (Ed.). *Human Mental Workload*. North-Holland, 1988, (Advances in Psychology, v. 52). p. 139 – 183. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0166411508623869>.

HVORECKý, J.; DRLíK, M.; MUNK, M. The effect of visual query languages on the improvement of information retrieval skills. *Procedia - Social and Behavioral Sciences*, v. 2, n. 2, p. 717 – 723, 2010. ISSN 1877-0428. Innovation and Creativity in Education. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877042810001308>.

JAAKKOLA, H.; THALHEIM, B. Visual sql – high-quality er-based query treatment. In: _____. *Conceptual Modeling for Novel Application Domains: ER 2003 Workshops ECOMO, IWCMQ, AOIS, and XSDM, Chicago, IL, USA, October 13, 2003. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 129–139. ISBN 978-3-540-39597-3. Disponível em: <http://dx.doi.org/10.1007/978-3-540-39597-3_13>.

KARDAS, G.; GOMEZ-SANZ, J. J. Special issue on model-driven engineering of multi-agent systems in theory and practice. *Computer Languages, Systems & Structures*, v. 50, n. Supplement C, p. 140 – 141, 2017. ISSN 1477-8424. Disponível em: <http://www.sciencedirect.com/science/article/pii/S147784241730101X>.

KAWASH, J. Complex quantification in structured query language (sql): A tutorial using relational calculus. *Journal of Computers in Mathematics and Science Teaching*, v. 23, n. 2, p. 169–190, 2004.

KEIM, D. A.; LUM, V. Visual query specification in a multimedia database system. In: *Proceedings of the 3rd Conference on Visualization '92*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1992. (VIS '92), p. 194–201. ISBN 0-8186-2896-0. Disponível em: <http://dl.acm.org/citation.cfm?id=949685.949722>.

KELLY, S.; TOLVANEN, J. *Domain-Specific Modeling: Enabling Full Code Generation*. [S.l.]: Wiley, 2008. (Wiley - IEEE). ISBN 9780470249253.

KERAMOPOULOS, E.; POUYIOUTAS, P.; SADLER, C. Goql, a graphical query language for object-oriented database systems. In: *Proceedings of the 3rd Basque International Workshop on Information Technology (BIWIT '97)*. Washington, DC, USA: IEEE Computer Society, 1997. (BIWIT '97), p. 35–. ISBN 0-8186-8049-0. Disponível em: <http://dl.acm.org/citation.cfm?id=523989.791426>.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. [S.l.], 2007.

KITCHENHAM, B.; MADEYSKI, L.; BUDGEN, D.; KEUNG, J.; BRERETON, P.; CHARTERS, S.; GIBBS, S.; POHTHONG, A. Robust statistical methods for empirical software engineering. *Empirical Software Engineering*, Springer, v. 22, n. 2, p. 579–630, 2017.

KLUG, A. C. Abe: A query language for constructing aggregates-by-example. In: *Proceedings of the 1st LBL Workshop on Statistical Database Management*. Berkeley, CA, US: Lawrence Berkeley Laboratory, 1981. (SSDBM'81), p. 190–205. ISBN 1-55555-222-X. Disponível em: <http://dl.acm.org/citation.cfm?id=1115950.1115976>.

KOLOVOS, D.; ROSE, L.; GARCIA-DOMINGUEZ, A. et al. The epsilon validation language', in. *The Epsilon Book*, p. 57–76, 2017.

KWAK, J.-C.; MOON, S. Two-dimensional specification of queries in object-oriented databases. *Microprocessing and Microprogramming*, v. 41, n. 3, p. 227 – 244, 1995. ISSN 0165-6074. Disponível em: <http://www.sciencedirect.com/science/article/pii/0165607495000059>.

LONGO, L. A defeasible reasoning framework for human mental workload representation and assessment. *Behaviour & Information Technology*, Taylor & Francis, v. 34, n. 8, p. 758–786, 2015.

MAXWELL, J. A. *Qualitative research design: An interactive approach*. [S.l.]: Sage publications, 2012. v. 41.

MOFFATT, C. *Visual Representation of SQL Joins*. 2009. Https://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins. Accessed: 2020-01-01.

MOHAN, L.; KASHYAP, R. L. A visual query language for graphical interaction with schema-intensive databases. *IEEE Trans. on Knowl. and Data Eng.*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 5, n. 5, p. 843–858, out. 1993. ISSN 1041-4347. Disponível em: <http://dx.doi.org/10.1109/69.243513>.

MOODY, D. The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, v. 35, n. 6, p. 756–779, Nov 2009. ISSN 0098-5589.

MUN-KEW, L.; BOON-SIONG, C.; CHUN-HONG, K.; JYH-JANG, L.; NARASIMHALU, D. The implementation of a visual language interface for an object-oriented multimedia database system. *Journal of Visual Languages & Computing*, v. 1, n. 3, p. 275 – 289, 1990. ISSN 1045-926X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1045926X05800102>.

MURRAY, N. S.; PATON, N. W.; GOBLE, C. A.; BRYCE, J. Kaleidoquery—a flow-based visual language and its evaluation. *Journal of Visual Languages & Computing*, v. 11, n. 2, p. 151 – 189, 2000. ISSN 1045-926X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1045926X99901507>.

NICKERSON, J. V. Visual programming: Limits of graphic representation. In: IEEE. *Visual Languages, 1994. Proceedings., IEEE Symposium on*. [S.l.], 1994. p. 178–179.

NIELSEN, J. 10 usability heuristics for user interface design. *Nielsen Norman Group*, v. 1, n. 1, 1995.

OMG. *Meta Object Facility (MOF) Core Specification, Version 2.0*. 2006. Http://www.omg.org/spec/MOF/. Accessed: 2017-02-14.

OMG. *Object Constraint Language, v2.4*. 2014. Http://www.omg.org/spec/OCL/2.4/PDF. Accessed: 2016-05-05.

PAULO, J. *Um Catálogo de Regras de Boa Formação para Consultas SQL*. 2017. Monografia (Bacharel em Ciência da Computação), UFPE (Universidade Federal de Pernambuco), Recife, Brasil.

RAMOS, H. B. Design and implementation of a graphical sql with generic capabilities. In: COOPER, R. (Ed.). *Interfaces to Database Systems (IDS92)*. London: Springer London, 1993. p. 74–91. ISBN 978-1-4471-3423-7.

SANTUCCI, G.; SOTTILE, P. A. Query by diagram: a visual environment for querying databases. *Software: Practice and Experience*, Wiley Online Library, v. 23, n. 3, p. 317–340, 1993.

SOCKUT, G. H.; BURNS, L. M.; MALHOTRA, A.; WHANG, K.-Y. Graqula: A graphical query language for entity-relationship or relational databases. *Data & Knowledge Engineering*, v. 11, n. 2, p. 171 – 202, 1993. ISSN 0169-023X. Disponível em: <http://www.sciencedirect.com/science/article/pii/0169023X93900049>.

SONG, E.; YIN, S.; RAY, I. Using uml to model relational database operations. *Computer Standards & Interfaces*, v. 29, n. 3, p. 343 – 354, 2007. ISSN 0920-5489. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0920548906000717>.

STACKOVERFLOW. *Stack Overflow Developer Survey Results 2018*. 2018. Https://insights.stackoverflow.com/survey/2018. Accessed: 2018-12-14.

TAN, K. P.; CHAN, H. C.; SIAU, K. L. A graphical knowledge level approach for user-database interaction. In: *Proceedings., Fourteenth Annual International Computer Software and Applications Conference*. [S.l.: s.n.], 1990. p. 453–458.

TANSEL, A. U.; ARKUN, M. E.; OZSOYOGLU, G. Time-by-example query language for historical databases. *IEEE Transactions on Software Engineering*, v. 15, n. 4, p. 464–478, April 1989. ISSN 0098-5589.

THUSOO, A.; SARMA, J. S.; JAIN, N.; SHAO, Z.; CHAKKA, P.; ANTHONY, S.; LIU, H.; WYCKOFF, P.; MURTHY, R. Hive: A warehousing solution over a map-reduce framework. *Proc. VLDB Endow.*, VLDB Endowment, v. 2, n. 2, p. 1626–1629, ago. 2009. ISSN 2150-8097. Disponível em: <https://doi.org/10.14778/1687553.1687609>.

WARE, C. Chapter 9 - images, words, and gestures. In: WARE, C. (Ed.). *Information Visualization (Second Edition)*. Second edition. San Diego: Academic Press, 2004, (Interactive Technologies). p. 297 – 316. ISBN 978-1-55860-819-1. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9781558608191500121>.

WHANG, K. Y.; MALHOTRA, A.; SOCKUT, G. H.; BURNS, L.; CHOI, K. S. Two-dimensional specification of universal quantification in a graphical database query language. *IEEE Transactions on Software Engineering*, v. 18, n. 3, p. 216–224, Mar 1992. ISSN 0098-5589.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. *Experimentation in software engineering*. [S.l.]: Springer Science & Business Media, 2012.

ZHANG, G.; CHU, W. W.; MENG, F.; KONG, G. Query formulation from high-level concepts for relational databases. In: *Proceedings User Interfaces to Data Intensive Systems*. [S.l.: s.n.], 1999. p. 64–74.

ZLOOF, M. M. et al. Query by example. In: *AFIPS National Computer Conference*. [S.l.: s.n.], 1975. v. 44, p. 431–438.

APÊNDICE A – CÓDIGO EMFATIC

Listagem A.1 – Código Emfatic

```
1  @namespace(uri="dsql", prefix="dsql")
2  @gmf
3  package dsql;
4
5  @gmf.diagram(onefile="true", diagram.extension="dsql")
6  class QueryDiagram {
7    val Query[*] queries;
8    val Link[*] links;
9    val LogicalOperation[*] logicalOperations;
10   val ConditionalExpression[*] conditionalExpressions;
11 }
12
13 @gmf.node(resizable="false", label="name", label.icon="false", border.width
       ="1", border.color="0,0,0", margin="3", figure="rectangle")
14 class Query {
15   attr String name;
16   attr int limitOffset;
17   attr int limitRowCount;
18   attr ValueRepetition valueRepetition;
19
20   @gmf.compartment(layout="list")
21   val ColumnReference[*] select;
22
23   @gmf.compartment
24   val SourceExpression[*] from;
25
26   @gmf.compartment(layout="list")
27   val Expression[0..1] whereHaving;
28 }
29
30 @gmf.node(resizable="false", label="name", label.icon="false", border.width
       ="1", border.color="255,255,255", margin="3", figure="rectangle")
31 abstract class ColumnReference {
32   attr String name;
33   attr int groupPosition;
34   attr int sortPosition;
35   attr String alias;
36   attr SortSpecification sortSpecification;
37   attr AggregationFunction aggregationFunction;
38 }
39
```

```
40 abstract class SourceExpression { }
41
42 @gmf.node(resizable="false", label="name", label.icon="false", border.width
     ="1", border.color="255,255,255", margin="3", figure="rectangle")
43 class Source extends SubqueryOrigin, SourceExpression {
44   attr SourceType type;
45   attr String name;
46   attr String alias;
47 }
48
49 abstract class Expression extends Operand, SourceExpression,
     ColumnReference {
50
51 }
52
53 @gmf.node(label="operation", border.color="0,0,0", border.style="dash",
     margin="3", figure="rectangle")
54 abstract class Operation extends Expression {
55   attr boolean negation;
56 }
57
58 abstract class Operand { }
59
60 class UnaryOperation extends Operation {
61   attr UnaryOp operation;
62
63   @gmf.compartment(layout="list")
64   val Operand[1] operand1;
65 }
66
67 class BinaryOperation extends Operation {
68   attr BinaryOp operation;
69
70   @gmf.compartment(layout="list")
71   val Operand[1] operand1;
72   @gmf.compartment(layout="list")
73   val Operand[1] operand2;
74 }
75
76 class TernaryOperation extends Operation {
77   attr TernaryOp operation;
78
79   @gmf.compartment(layout="list")
80   val Operand[1] operand1;
81   @gmf.compartment(layout="list")
82   val Operand[1] operand2;
83   @gmf.compartment(layout="list")
```

```
84    val Operand[1] operand3;
85  }
86
87
88  @gmf.node(label="type", resizable="false", label.icon="false", border.width
        ="1", border.color="0,0,0", margin="3", figure="rectangle")
89  class LogicalOperation  {
90    attr LogicalOperationType type;
91    attr boolean negation;
92
93    @gmf.compartment(layout="list")
94    val Expression[1..*] expressions;
95  }
96
97  abstract class ConditionalExpressionOrigin {
98    ref CaseLink[1]#source caseLink;
99  }
100
101 @gmf.node(label="label", label.readOnly="true", resizable="false", label.
        icon="false", border.width="1", border.color="0,0,0", margin="3", figure
        ="rectangle")
102 abstract class ConditionalExpression {
103 }
104
105 class When extends ConditionalExpression, ConditionalExpressionOrigin {
106   attr String label;
107   @gmf.compartment(layout="list")
108   val Expression[1] condition;
109
110   @gmf.compartment(layout="list")
111   val Expression[1] result;
112 }
113
114 class Else extends ConditionalExpression {
115   attr String label;
116
117   @gmf.compartment(layout="list")
118   val Expression[1] result;
119 }
120
121
122 abstract class SubqueryOrigin {
123   ref SubqueryLink[1]#target subqueryLinkTarget;
124 }
125
126 class Column extends SubqueryOrigin, ConditionalExpressionOrigin,
        Expression {
```

```
127   attr ColumnType type;
128   ref LogicalLink[1]#source logicalLink;
129 }
130
131 @gmf.link(source="origin", target="target", incoming="true", color="0,0,0",
        width="1")
132 abstract class Link {
133 }
134
135 @gmf.link(style="dash", target.decoration="filledclosedarrow", label="
      setOperator")
136 class SetLink extends Link {
137   attr ValueRepetition valueRepetition;
138   attr SetOperator setOperator;
139   ref Query[1] origin;
140   ref Query[1] target;
141 }
142
143 @gmf.link(style="dash", target.decoration="filledclosedarrow", label="
      joinType")
144 class JoinLink extends Link {
145   attr String foreignKeyName;
146   attr JoinType joinType;
147   ref Source[1] origin;
148   ref Source[1] target;
149 }
150
151 @gmf.link(style="dash")
152 class JoinConditionLink extends Link {
153   ref JoinLink[1] origin;
154   ref Expression[1] target;
155 }
156
157 @gmf.link(style="solid", label="operation")
158 class SubqueryLink extends Link {
159   attr BinaryOp operation;
160   ref Query[1] target;
161   ref SubqueryOrigin[1]#subqueryLinkTarget origin;
162 }
163
164 @gmf.link(style="solid", target.decoration="arrow")
165 class LogicalLink extends Link {
166   ref Column[1]#logicalLink origin;
167   ref LogicalOperation[1] target;
168 }
169
170 @gmf.link(style="solid", target.decoration="closedarrow")
```

```
171 class CaseLink extends Link {
172   ref ConditionalExpressionOrigin[1]#caseLink origin;
173   ref ConditionalExpression[1] target;
174 }
175
176 enum SetOperator {
177   Intersect = 0;
178   Union = 1;
179   Except = 2;
180   ExclusiveUnion = 3;
181 }
182
183 enum JoinType {
184   Inner = 0;
185   FullOuter = 1;
186   LeftOuter = 2;
187   RightOuter = 3;
188   FullOuterExcluding = 4;
189   LeftOuterExcluding = 5;
190   RightOuterExcluding = 6;
191   Semi = 7;
192   Anti = 8;
193   Division = 9;
194 }
195
196 enum LogicalOperationType {
197   And = 0;
198   Or = 1;
199 }
200
201 enum ColumnType {
202   Field = 0;
203   Literal = 1;
204   FreeExpression = 2;
205   LogicalConnective = 3;
206   ConditionalExpression = 4;
207   Subquery = 5;
208 }
209
210 enum SourceType {
211   Table = 0;
212   Subquery = 1;
213 }
214
215 enum ValueRepetition {
216   All = 0;
217   Distinct = 1;
```

```
}

enum SortSpecification {
  Ascending = 0;
  Descending = 1;
}

enum AggregationFunction {
  None = 0;
  AVG = 1;
  MAX = 2;
  MIN = 3;
  SUM = 4;
  COUNT = 5;
}

enum UnaryOp {
  Exists = 0;
  IsNull = 1;
  Unique = 2;
}

enum BinaryOp {
  Equals = 0;
  NotEquals = 1;
  GreaterThan = 2;
  GreaterThanOrEquals = 3;
  LessThan = 4;
  LessThanOrEquals = 5;
  Addition = 6;
  Subtraction = 7;
  Multiplication = 8;
  Divison = 9;
  Concatenation = 10;
  In = 11;
  Like = 12;
  Similar = 13;
  Overlaps = 14;
  Any = 15;
  AnyEquals = 16;
  AnyNotEquals = 17;
  AnyGreaterThan = 18;
  AnyGreaterThanOrEquals = 19;
  AnyLessThan = 20;
  AnyLessThanOrEquals = 21;
  All = 22;
  AllEquals = 23;
```

```
265     AllNotEquals = 24;
266     AllGreaterThan = 25;
267     AllGreaterThanOrEquals = 26;
268     AllLessThan = 27;
269     AllLessThanOrEquals = 28;
270     None = 99;
271 }
272 
273 enum TernaryOp {
274     Between = 0;
275 }
```

APÊNDICE B – TERMO DE CONSENTIMENTO

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

(PARA MAIORES DE 18 ANOS OU EMANCIPADOS - Resolução 466/12)

Convidamos o (a) Sr. (a) para participar como voluntário (a) da pesquisa "Uma Linguagem Visual para Diagramar Consultas SQL", que está sob a responsabilidade do pesquisador Edson Alves da Silva, residente à Avenida Noraci Pedrosa, 150, Apt 1002, Antares, Maceió-AL, 57083-060 – Telefone 81-99962-1093 e e-mail para contato eas4@cin.ufpe.br.

Caso este Termo de Consentimento contenha informações que não lhe sejam compreensíveis, as dúvidas podem ser sanadas com a pessoa que está lhe entrevistando e apenas ao final, quando todos os esclarecimentos forem dados, caso concorde com a realização do estudo pedimos que assine ao final deste documento, que está em duas vias, uma via lhe será entregue e a outra ficará com o pesquisador responsável.

Caso não concorde, não haverá penalização alguma por não participar ou desistir do experimento, bem como será possível retirar o consentimento a qualquer momento, também sem nenhuma penalidade.

INFORMAÇÕES SOBRE A PESQUISA:

O objetivo do experimento é analisar se a compreensão de consultas em DSQL tem o potencial de ser tão eficiente quanto em SQL.

Os dados coletados através de entrevistas nesta pesquisa serão confidenciais e serão divulgadas apenas em eventos ou publicações científicas, não havendo identificação dos voluntários, a não ser entre os responsáveis pelo estudo, sendo assegurado o sigilo sobre a sua participação. Os dados coletados nesta pesquisa, através de entrevistas ficarão armazenados em arquivos digitais, sob a responsabilidade do pesquisador, pelo período aproximado de 1 ano.

Nada lhe será pago e nem será cobrado para participar desta pesquisa, pois a aceitação é voluntária, mas fica também garantida a indenização em casos de danos, comprovadamente decorrentes da participação na pesquisa, caso alguma decisão judicial ou extrajudicial se aplique.

Em caso de dúvidas relacionadas aos aspectos éticos deste estudo, você poderá consultar o Comitê de Ética em Pesquisa Envolvendo Seres Humanos da UFPE no endereço: (Avenida da Engenharia s/n – 1º Andar, sala 4 - Cidade Universitária, Recife-PE, CEP: 50740-600, Tel.: (81) 2126.8588 – e-mail: cepccs@ufpe.br).

_____

(assinatura do pesquisador)

CONSENTIMENTO DA PARTICIPAÇÃO DA PESSOA COMO VOLUNTÁRIO (A)

Eu, _____, CPF _____, abaixo assinado, após a leitura (ou a escuta da leitura) deste documento e de ter tido a oportunidade de conversar e ter esclarecido as minhas dúvidas com o pesquisador responsável, concordo em participar do estudo Uma Linguagem Visual para Diagramar Consultas SQL, como voluntário(a). Fui devidamente informado(a) e esclarecido(a) pelo pesquisador sobre a pesquisa, os procedimentos nela envolvidos, assim como os possíveis riscos e benefícios decorrentes de minha participação. Foi-me garantido que posso retirar o meu consentimento a qualquer momento, sem que isto leve a qualquer penalidade.

Recife, 01 de janeiro de 2019

_____

(assinatura da pessoa participante)

APÊNDICE C – CURSO DE SQL E DSQL

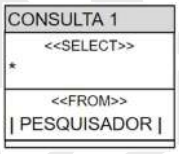
DML - Consultando Dados em Tabelas

- **Exemplos**:

--CONSULTA 1 - PROJETAR TODAS AS INFORMAÇÕES DOS PESQUISADORES

SELECT *

FROM PESQUISADOR ;



DML - Consultando Dados em Tabelas

- **Exemplos**:

-- CONSULTA 2 - PROJETAR A MÉDIA DAS NOTAS DOS ARTIGOS

SELECT AVG (NOTA)

FROM ARTIGO;

DML - Consultando Dados em Tabelas

- **Exemplos**:

/* CONSULTA 3 - PROJETAR TODOS OS ARTIGOS(TÍTULOS) E SUAS NOTAS COM -1 PONTO */

SELECT TITULO, NOTA – 1

FROM ARTIGO;

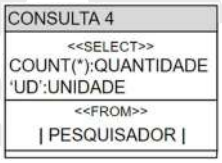
DML - Consultando Dados em Tabelas

- **Exemplos**:

/* CONSULTA 4 - PROJETAR A QUANTIDADE DE PESQUISADORES CADASTRADOS*/

SELECT COUNT (*) AS QUANTIDADE, 'UD' AS UNIDADE

FROM PESQUISADOR;



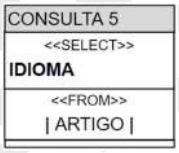
DML - Consultando Dados em Tabelas

- **Exemplos**:

-- CONSULTA 5 - PROJETAR TODOS OS IDIOMAS DOS ARTIGOS

SELECT ALL IDIOMA ***-- ou SELECT IDIOMA***

FROM ARTIGO;



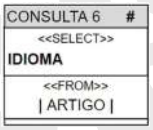
DML - Consultando Dados em Tabelas

- **Exemplos**:

/* CONSULTA 6 - PROJETAR TODOS OS IDIOMAS, SEM DUPLICATAS, DOS ARTIGOS */

SELECT DISTINCT IDIOMA

FROM ARTIGO;



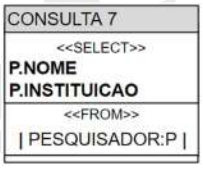
DML - Consultando Dados em Tabelas

- **Exemplos**:

/* CONSULTA 7 - PROJETAR TODOS OS NOMES E INSTITUIÇÕES DA TABELA PESQUISADOR */

SELECT P.NOME, P.INSTITUICAO

FROM PESQUISADOR P;



DML - Consultando Dados em Tabelas

- **Exemplos**:

/* CONSULTA 8 - PROJETAR TODOS OS TÍTULOS DOS ARTIGOS E SUAS NOTAS COM –1 PONTO */

SELECT TITULO, NOTA -1 AS REDUCAO

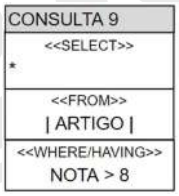
FROM ARTIGO;

DML - Consultando Dados em Tabelas

- **Exemplos**:

--CONSULTA 9 - PROJETAR ARTIGOS COM NOTA ACIMA DE 8

```
SELECT *
FROM ARTIGO
WHERE NOTA > 8;
```

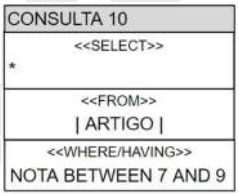


DML - Consultando Dados em Tabelas

- **Exemplos**:

-- CONSULTA 10 - PROJETAR ARTIGOS COM NOTA ENTRE 7 e 9

```
SELECT *
FROM ARTIGO
WHERE NOTA BETWEEN 7 AND 9;
```



DML - Consultando Dados em Tabelas

- **Exemplos**:

/* CONSULTA 11 - PROJETAR OS EVENTOS QUE OCORRERAM EM 2010, 2012 OU 2014 */

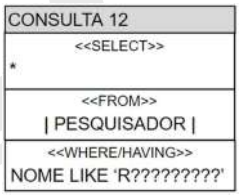
```
SELECT *
FROM  EVENTO
WHERE ANO IN (2010, 2012, 2014);
```

DML - Consultando Dados em Tabelas

- **Exemplos**

/* CONSULTA 12 - PROJETAR TODOS OS PESQUISADORES CUJO NOME TENHA 10 CARACTERES E INICIE COM R */

```
SELECT *
FROM PESQUISADOR
WHERE NOME LIKE 'R??????????';
```



DML - Consultando Dados em Tabelas

- **Exemplos**

/*CONSULTA 13 - PROJETAR TODOS OS ARTIGOS QUE TENHAM BANCO DE DADOS NO SEU TÍTULO */

```
SELECT *
FROM ARTIGO
WHERE TITULO LIKE '%BANCO DE DADOS%';
```

DML - Consultando Dados em Tabelas

- **Exemplos**

/*CONSULTA 14 - PROJETAR TODOS OS ARTIGOS QUE ESTÃO SEM NOTA DEFINIDA */

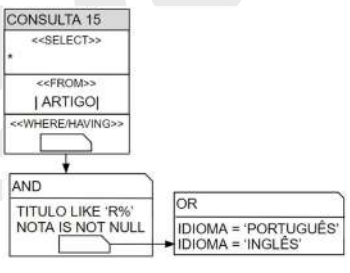
```
SELECT *
FROM ARTIGO
WHERE A IS NULL;
```

DML - Consultando Dados em Tabelas

- **Exemplo**

/* CONSULTA 15 - PROJETAR TODOS OS ARTIGOS QUE INICIAM COM R, ESTÃO COM NOTA DEFINIDA E FORAM ESCRITOS EM PORTUGUÊS OU INGLÊS*/

```
SELECT *
FROM ARTIGO
WHERE TITULO LIKE 'R%' AND
        NOTA IS NOT NULL  AND
        (IDIOMA = 'PORTUGUÊS' OR
         IDIOMA = 'INGLÊS');
```

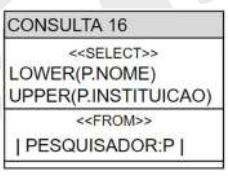


DML - Consultando Dados em Tabelas

- **Exemplo**

/* CONSULTA 16 - PROJETAR , EM LETRAS MINÚSCULAS, OS NOMES E, EM LETRAS MAIÚSCULAS, AS INSTITUIÇÕES DA TABELA PESQUISADOR */

```
SELECT LOWER(P.NOME), UPPER (P.INSTITUICAO)
FROM PESQUISADOR P;
```



DML - Consultando Dados em Tabelas

- **Exemplo** (Funções para manipular strings)

CONCAT('Hello','World');
RESULTADO: HelloWorld

SUBSTR/SUBSTRING('HelloWorld',1,5)
RESULTADO Hello

LENGTH/LEN('HelloWorld')
RESULTADO: 10

INSTR('HelloWorld', 'W')
CHARINDEX('W', 'HelloWorld')
RESULTADO: 6

LPAD(99999, 10, '*') -- Só ORACLE
RESULTADO: *****99999

INITCAP('teste') -- Só ORACLE
RESULTADO: Teste

RPAD(99999, 10, '*') -- Só ORACLE
RESULTADO: 99999*****

REPLACE('Jack and Jue', 'J', 'Bl')
RESULTADO: Black and Blue

TRIM(' um teste doido ')
RESULTADO: [um teste doido]

TRIM('*' FROM '**teste**')
RESULTADO: [teste]

DML - Consultando Dados em Tabelas

- **Exemplo** (Funções para manipular strings)

/* *CONSULTA 17* - PROJETAR A CONCATENAÇÃO DE DUAS STRINGS E MOSTRÁ-LAS EM MAIÚSCULO */

```
SELECT 'Hello World' ORIGINAL, UPPER(CONCAT('Hello World  ', 'Teste') ) MODIFICADO
FROM DUAL;
```

CONSULTA 17
<<SELECT>>
'HELLO WORLD':ORIGINAL UPPER(CONCAT('HELLO WORLD', 'TESTE')):MODIFICADOR
<<FROM>>
| DUAL |

```
ORIGINAL          MODIFICADO
----------------- ------------------
Hello World       HELLO WORLD TESTE
```


DML - Consultando Dados em Tabelas

- **Exemplo** (Função para converter string em número)

-- *CONSULTA 18* - PROJETAR A SOMA DE UM NÚMERO E UMA STRING

```
SELECT 100+TO_NUMBER('100') -- (ou CAST('100' as INT) no SQL Server)
FROM DUAL;
```

CONSULTA 18
<<SELECT>>
100+TO_NUMBER('100')
<<FROM>>
| DUAL |

DML - Consultando Dados em Tabelas

- **Exemplo** (Funções para manipular de números)

Arredondar com 2 casas
ROUND (45.926, 2)
RESULTADO 45.93

Eliminar a terceira casa decimal
TRUNC(45.926, 2) -- Só ORACLE
RESULTADO 45.92

Obter o resto da divisão
MOD(1600,300) -- Só ORACLE
RESULTADO 100

Obter a dezena
ROUND(45.923, -1)
RESULTADO 50

DML - Consultando Dados em Tabelas

- **Exemplo** (Manipulando datas)

-- *CONSULTA 19* - PROJETAR A DATA DO SISTEMA

```
SELECT SYSDATE -- (ou GETDATE())
FROM DUAL
```

CONSULTA 19
<<SELECT>>
SYSDATE
<<FROM>>
| DUAL |

DML - Consultando Dados em Tabelas

- **Exemplo** (Manipulando datas)

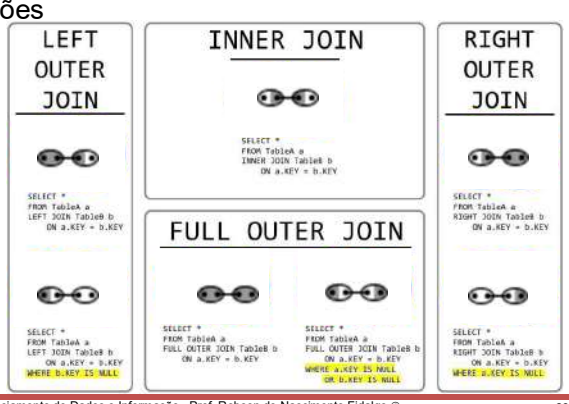
-- *CONSULTA 20* - PROJETAR O DIA, MÊS E ANO DE DATA DO SISTEMA

```
SELECT DAY(SYSDATE), MONTH(SYSDATE), YEAR(SYSDATE),
FROM DUAL;
```

CONSULTA 20
<<SELECT>>
DAY(SYSDATE) MONTH(SYSDATE) YEAR(SYSDATE)
<<FROM>>
| DUAL |

DML - Consultando Dados em Tabelas

- Junções



DML - Consultando Dados em Tabelas

- **Exemplo** (junção interna – Old School)

/* *CONSULTA 21* - PROJETAR O TÍTULO DE TODOS OS ARTIGOS PUBLICADOS E A SIGLA DE SEUS RESPECTIVOS EVENTOS - SEM USAR "INNER JOIN" */

```
SELECT A.TITULO, E.SIGLA
FROM ARTIGO A, EVENTO E
WHERE  A.COD = E.COD;
```

CONSULTA 21
<<SELECT>>
A.TITULO E.SIGLA
<<FROM>>
| ARTIGO:A | | EVENTO:E |
<<WHERE/HAVING>>
A.COD = E.COD

DML - Consultando Dados em Tabelas

- **Exemplo** (junção interna – New School)

/ CONSULTA 22 - PROJETAR O TÍTULO DE TODOS OS ARTIGOS PUBLICADOS E A SIGLA DOS SEUS RESPECTIVOS EVENTOS - USANDO "INNER JOIN" */*

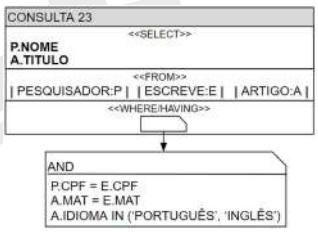
```
SELECT A.TITULO, E.SIGLA
FROM ARTIGO A INNER JOIN
        EVENTO E ON A.COD = E.COD;
```

DML - Consultando Dados em Tabelas

- **Exemplo** (junção interna encadeada – Old School)

/ CONSULTA 23 - PROJETAR OS NOMES DE TODOS OS PESQUISADORES E OS TÍTULOS DE SEUS RESPECTIVOS ARTIGOS PUBLICADOS NOS IDIOMAS PORTUGUÊS OU INGLÊS - SEM USAR "INNER JOIN" */*

```
SELECT P.NOME,A.TITULO
FROM PESQUISADOR, ESCREVE, ARTIGO
WHERE P.CPF =  E.CPF AND
        E.MAT =  A.MAT AND
        A.IDIOMA IN ('PORTUGUÊS','INGLÊS') ;
```



DML - Consultando Dados em Tabelas

- **Exemplo** (junção interna encadeada – New School)

/ CONSULTA 24 - PROJETAR OS NOMES DE TODOS OS PESQUISADORES E OS TÍTULOS DOS SEUS RESPECTIVOS ARTIGOS PUBLICADOS NOS IDIOMAS PORTUGUÊS OU INGLÊS - USANDO "INNER JOIN" */*

```
SELECT P.NOME,A.TITULO
FROM PESQUISADOR P INNER JOIN
   ESCREVE E ON P.CPF = E.CPF INNER JOIN
   ARTIGO A ON E.MAT = A.MAT
WHERE A.IDIOMA IN ('PORTUGUÊS','INGLÊS') ;
```

DML - Consultando Dados em Tabelas

- **Exemplo** (junção externa à esquerda)

/ CONSULTA 25 - PROJETAR OS TÍTULOS DE TODOS OS ARTIGOS E SEUS RESPECTIVOS EVENTOS (SIGLAS) */*

```
SELECT A.TITULO, E.SIGLA
FROM ARTIGO A LEFT OUTER JOIN
        EVENTO E ON A.COD = E.COD;
```

DML - Consultando Dados em Tabelas

- **Exemplo** (junção externa à direita)

/ CONSULTA 26 - PROJETAR TODOS OS ARTIGOS (TÍTULOS) E SEUS RESPECTIVOS EVENTOS (SIGLAS) */*

```
SELECT A.TITULO, E.SIGLA
FROM EVENTO E RIGHT OUTER JOIN
        ARTIGO A ON A.COD = E.COD;
```

DML - Consultando Dados em Tabelas

- **Exemplo** (junção externa completa)

/ CONSULTA 27 - PROJETAR OS ARTIGOS (TÍTULOS) E SEUS RESPECTIVOS EVENTOS (SIGLAS), INCLUINDO OS ARTIGOS NÃO PUBLICADOS E OS EVENTOS SEM ARTIGOS */*

```
SELECT A.TITULO, E.SIGLA
FROM ARTIGO A FULL OUTER JOIN
        EVENTO E ON A.COD = E.COD;
```

DML - Consultando Dados em Tabelas

- **Exemplo** (junção externa à esquerda exclusiva)

/ CONSULTA 28 - PROJETAR OS EVENTO (SIGLAS) SEM ARTIGOS PUBLICADOS (NULL PARA TÍTULO DO ARTIGO) - USANDO "LEFT OUTER JOIN" */*

```
SELECT E.SIGLA, A.TITULO
FROM EVENTO E LEFT OUTER JOIN
        ARTIGO A ON E.COD = A.COD
WHERE A.COD IS NULL;
```

DML - Consultando Dados em Tabelas

- **Exemplo** (junção externa à direita exclusiva)

/ CONSULTA 29 - PROJETAR OS EVENTO (SIGLAS) SEM ARTIGOS PUBLICADOS (NULL PARA TÍTULO DO ARTIGO) - USANDO "RIGTH OUTER JOIN" */*

```
SELECT E.SIGLA, A.TITULO
FROM ARTIGO A RIGHT OUTER JOIN
        EVENTO E ON E.COD = A.COD
WHERE A.COD IS NULL;
```

DML - Consultando Dados em Tabelas

- **Exemplo** (junção externa completa exclusiva)

/* CONSULTA 30 - PROJETAR OS ARTIGOS (TÍTULOS) NÃO PUBLICADOS E OS EVENTOS (SIGLAS) SEM ARTIGOS - USANDO "FULL OUTER JOIN" */

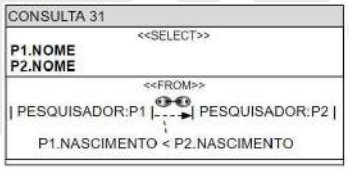
```
SELECT A.TITULO, E.SIGLA
FROM ARTIGO A FULL OUTER JOIN
        EVENTO E ON A.COD = E.COD
WHERE  A.COD IS NULL OR
        E.COD IS NULL;
```

DML - Consultando Dados em Tabelas

- **Exemplo** (auto-junção com theta join/non-equi join)

/* CONSULTA 31 - PARA CADA PESQUISADOR, PROJETAR SEU NOME E OS NOMES DOS PESQUISADORES MAIS NOVOS DO QUE ELE */

```
SELECT P1.NOME, P2.NOME
FROM PESQUISADOR P1 INNER JOIN
        PESQUISADOR P2 ON P1.NASCIMENTO < P2.NASCIMENTO
```

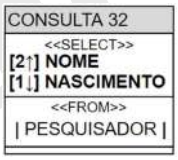


DML - Consultando Dados em Tabelas

- **Exemplo** (ordenando o resultado)

/* CONSULTA 32 - PROJETAR O NOME E O NASCIMENTO DOS PESQUISADORES EM ORDEM DECRESCENTE DO NASCIMENTO. PARA DATAS IGUAIS, CONSIDERAR A ORDEM ALFABÉTICA DO NOME DO PESQUISADOR */

```
SELECT NOME, NASCIMENTO
FROM PESQUISADOR
ORDER BY NASCIMENTO DESC, NOME ASC;
```



DML - Consultando Dados em Tabelas

- **Exemplo (agrupando/sumarizando o resultado)**

-- CONSULTA 33 - PROJETAR A MÉDIA DAS NOTAS DOS ARTIGOS POR EVENTO

```
SELECT E.SIGLA, AVG (A.NOTA) AS MEDIA_NOTA
FROM ARTIGO A INNER JOIN
        EVENTO E ON A.COD = E. COD
GROUP BY E.SIGLA;
```

DML - Consultando Dados em Tabelas

- **Exemplo (restringindo os grupos do resultado)**

/* CONSULTA 34 - PROJETAR AS SIGLAS DOS EVENTOS CUJAS MÉDIAS DAS NOTAS DOS ARTIGOS SÃO MAIORES DO QUE 8 */

```
SELECT E.SIGLA, AVG (A.NOTA)
FROM EVENTO E INNER JOIN
        ARTIGO A  ON E.COD = A. COD
GROUP BY E.SIGLA
HAVING AVG (A.NOTA) > 8;
```

DML - Consultando Dados em Tabelas

- **Exemplo (agrupando/sumarizando o resultado)**

/* CONSULTA 35 - PROJETAR OS NOMES DOS PESQUISADORES QUE PUBLICARAM MAIS DE 3 ARTIGOS */

```
SELECT P.NOME, COUNT(*)
FROM PESQUISADOR P INNER JOIN
        ESCREVE E ON P.CPF = E.CPF INNER JOIN
GROUP BY P.NOME
HAVING COUNT(A.TITULO) > 3;
```

DML - Consultando Dados em Tabelas

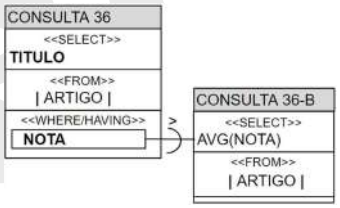
- Tipos de Subconsulta
  - Quanto a quantidade de linhas e colunas retornadas
    - ESCALAR → Retornam um único valor (i.e., uma única linha e coluna)
    - LINHA → Retornam várias colunas, mas apenas uma linha
    - TABELA→ Retornam uma ou mais colunas e múltiplas linhas
  - Quanto a dependências entre as subconsultas
    - SIMPLES
    - CORRELACIONADA
      - SEMI JOIN
      - ANTI JOIN (ou ANTI SEMI JOIN – negação do SEMI JOIN)

DML - Consultando Dados em Tabelas

- **Exemplo** (subconsulta simples e escalar)

/* CONSULTA 36 - PROJETAR OS TÍTULOS DOS ARTIGOS COM NOTA ACIMA DA MÉDIA */

```
SELECT TITULO
FROM ARTIGO
WHERE NOTA >
        (SELECT AVG (NOTA)
        FROM ARTIGO) ;
```

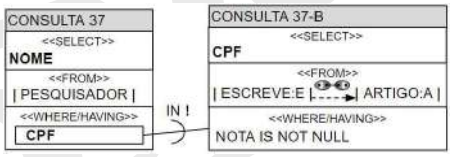


DML - Consultando Dados em Tabelas

- **Exemplo** (subconsulta simples e tabela)

/* CONSULTA 37 - PROJETAR OS NOMES DOS PESQUISADORES QUE POSSUEM ARTIGOS SEM NOTA - USANDO "IN" E "INNER JOIN") */

```
SELECT NOME
FROM PESQUISADOR
WHERE CPF NOT IN
      (SELECT CPF
       FROM ESCREVE E INNER JOIN
            ARTIGO A ON E.MAT =  A.MAT
       WHERE NOTA IS NOT NULL);
```

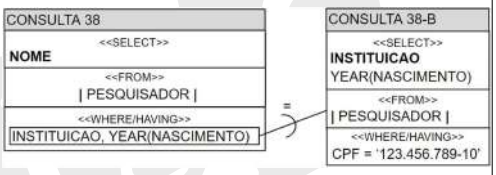


DML - Consultando Dados em Tabelas

- **Exemplo** (subconsulta simples e linha)

/* CONSULTA 38 - PROJETAR OS NOMES DOS PESQUISADORES QUE NASCERAM NO MESMO ANO E TRABALHAM NA MESMA INSTITUIÇÃO DO PESQUISADOR 123.456.789-10 */

```
SELECT NOME
FROM PESQUISADOR
WHERE (INSTITUICAO, YEAR(NASCIMENTO))  =
          (SELECT INSTITUICAO, (YEAR(NASCIMENTO))
          FROM PESQUISADOR
          WHERE CPF = 123.456.789-10);
```

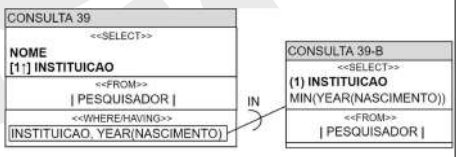


DML - Consultando Dados em Tabelas

- **Exemplo** (subconsulta simples e tabela)

/* CONSULTA 39 - POR INSTITUIÇÃO, PROJETAR O NOME E A INSTITUIÇÃO DOS PESQUISADORES MAIS VELHOS */

```
SELECT NOME, INSTITUICAO
FROM PESQUISADOR
WHERE (INSTITUICAO, YEAR(NASCIMENTO)) IN
   (SELECT INSTITUICAO, MIN(YEAR(NASCIMENTO))
    FROM PESQUISADOR
    GROUP BY INSTITUICAO)
ORDER BY INSTITUICAO;
```

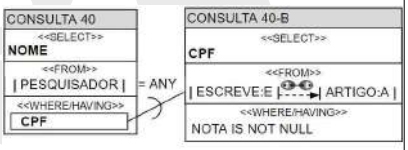


DML - Consultando Dados em Tabelas

- **Exemplo** (subconsulta simples e tabela)

/* CONSULTA 40 - PROJETAR OS NOMES DOS PESQUISADORES QUE POSSUEM ARTIGOS SEM NOTA - USANDO "= ANY", pois é equivalente ao "IN") */

```
SELECT NOME
FROM PESQUISADOR
WHERE CPF = ANY
   (SELECT CPF
   FROM ESCREVE E INNER JOIN
         ARTIGO A ON E.MAT = A.MAT
   WHERE NOTA IS NULL);
```

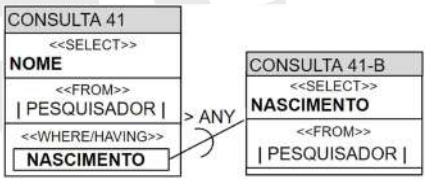


DML - Consultando Dados em Tabelas

- **Exemplo** (subconsulta simples e tabela)

/* CONSULTA 41 - PROJETAR OS NOMES DOS PESQUISADORES, EXCETO O DO MAIS VELHO */

```
SELECT NOME
FROM PESQUISADOR
WHERE NASCIMENTO > ANY
   (SELECT NASCIMENTO
    FROM PESQUISADOR)
```

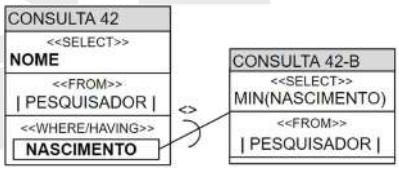


DML - Consultando Dados em Tabelas

- **Exemplo** (subconsulta simples e escalar)

/* CONSULTA 42 - PROJETAR OS NOMES DOS PESQUISADORES, EXCETO O DO MAIS IDOSO */

```
SELECT NOME
FROM PESQUISADOR
WHERE NASCIMENTO <>
   (SELECT MIM(NASCIMENTO)
    FROM PESQUISADOR)
```



DML - Consultando Dados em Tabelas

- **Exemplo** (subconsulta simples e tabela)

/* CONSULTA 43 - PROJETAR OS TÍTULOS DOS ARTIGOS COM NOTA MAIOR DO QUE TODOS OS ARTIGOS DO EVENTO 'SBBD' */

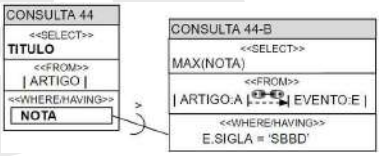
```
SELECT TITULO
FROM ARTIGO
WHERE NOTA > ALL
      (SELECT NOTA
       FROM ARTIGO A INNER JOIN
            EVENTO E A.COD = E.COD
       WHERE  E.SIGLA = 'SBBD');
```

DML - Consultando Dados em Tabelas

- **Exemplo** (subconsulta simples e escalar)

/* CONSULTA 44 - PROJETAR OS TÍTULOS DOS ARTIGOS COM NOTA MAIOR DO QUE TODOS OS ARTIGOS DO EVENTO 'SBBD' */

```
SELECT TITULO
FROM ARTIGO
WHERE NOTA >
      (SELECT MAX(NOTA)
       FROM ARTIGO A INNER JOIN
            EVENTO E  A.COD = E.COD
       WHERE E.SIGLA = 'SBBD');
```

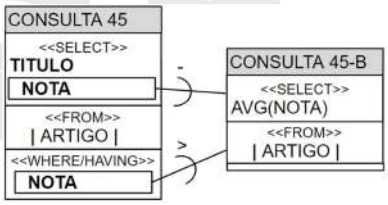


DML - Consultando Dados em Tabelas

- **Exemplo** (subconsultas simples e escalares – no SELECT/WHERE)

/* CONSULTA 45 - PROJETAR OS TÍTULOS DOS ARTIGOS COM NOTA ACIMA DA MÉDIA GERAL, MOSTRANDO QUANTO A NOTA ESTÁ ACIMA DA MÉDIA */

```
SELECT TÍTULO, NOTA – (SELECT AVG(A.NOTA) FROM ARTIGO A)
FROM ARTIGO
WHERE NOTA >
   ( SELECT AVG(A.NOTA)
     FROM ARTIGO A);
```

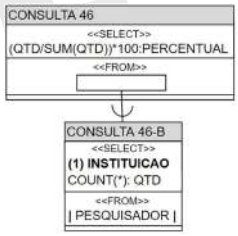


DML - Consultando Dados em Tabelas

- **Exemplo** (subconsulta simples e tabela – no FROM)

/* CONSULTA 46 - PROJETAR O PERCENTUAL DE PESQUISADORES POR INSTITUIÇÃO */

```
SELECT INSTITUICAO, (QTD/SUM(QTD))*100 AS PERCENTUAL
FROM
   (SELECT INSTITUICAO, COUNT(*) AS QTD
    FROM PESQUISADOR
    GROUP BY INSTITUICAO);
```

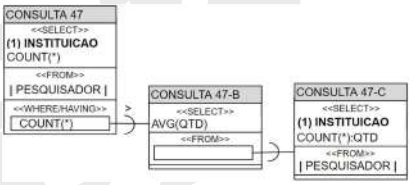


DML - Consultando Dados em Tabelas

- **Exemplo** (subconsulta simples e tabela – no HAVING)

/* CONSULTA 47 - PROJETAR QUAIS SÃO AS INSTITUIÇÕES QUE TÊM MAIS PESQUISADORES DO QUE A MÉDIA */

```
SELECT INSTITUICAO, COUNT(*)
FROM PESQUISADOR
GROUP BY INSTITUICAO
HAVING COUNT(*) >
        ( SELECT AVG(QTD)
          FROM (SELECT INSTITUICAO, COUNT (*) AS QTD
                FROM PESQUISADOR
                GROUP BY INSTITUICAO))
```

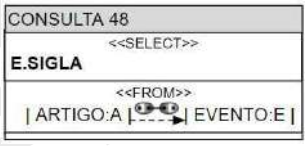


DML - Consultando Dados em Tabelas

- **Exemplo** (consulta com junção interna)

/* CONSULTA 48 - PROJETAR AS SIGLAS DOS EVENTOS COM ARTIGOS PUBLICADOS

```
SELECT E.SIGLA
FROM ARTIGO A INNER JOIN
        EVENTO E A.COD = E.COD;
```

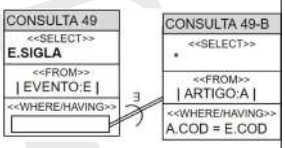


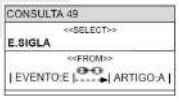
DML - Consultando Dados em Tabelas

- **Exemplo** (semi join a partir de subconsulta correlacionada e escalar)

/* CONSULTA 49 - PROJETAR AS SIGLAS DOS EVENTOS COM ARTIGOS PUBLICADOS – USANDO "EXISTS" */

```
SELECT E.SIGLA
FROM EVENTO E
WHERE  EXISTS
          (SELECT *
           FROM ARTIGO A
           WHERE A.COD = E.COD);
```





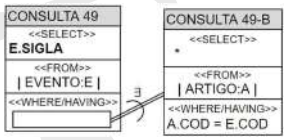
Notação alternativa

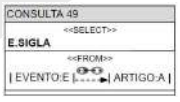
DML - Consultando Dados em Tabelas

- **Exemplo** (semi join a partir de subconsulta correlacionada e escalar)

ATENÇÃO:

SEMI JOIN = exibir os dados da tabela a esquerda a partir de uma comparação entre duas tabelas até encontrar o primeiro resultado verdadeiro. Isto é, **não há produto cartesiano, tampouco resultados repetidos.**





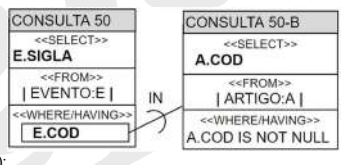
Notação alternativa

DML - Consultando Dados em Tabelas

- **Exemplo** (semi join a partir de subconsulta simples e tabela)

/* CONSULTA 50 - PROJETAR AS SIGLAS DOS EVENTOS COM ARTIGOS PUBLICADOS – USANDO "IN" */

```
SELECT E.SIGLA
FROM EVENTO E
WHERE  E. COD IN
          (SELECT A.COD
           FROM ARTIGO A
           WHERE A.COD IS NOT NULL);
```



ATENÇÃO:

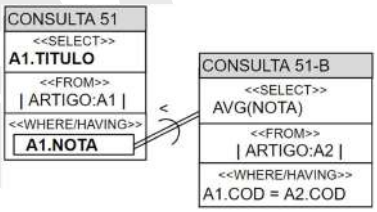
Subconsultas com EXIST usam índice, por isso são mais performática do que com IN.

DML - Consultando Dados em Tabelas

- **Exemplo** (consulta correlacionada)

/* CONSULTA 51 - PROJETAR OS TÍTULOS DOS ARTIGOS COM NOTA ABAIXO DA MÉDIA DOS ARTIGOS PUBLICADOS NO MESMO EVENTO*/

```
SELECT A1.TÍTULO
FROM ARTIGO A1
WHERE A1.NOTA <
   ( SELECT AVG(NOTA))
    FROM ARTIGO A2
    WHERE A1.COD = A2.COD);
```

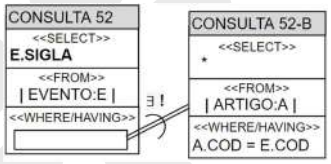


DML - Consultando Dados em Tabelas

- **Exemplo** (negação do semi join ou anti join)

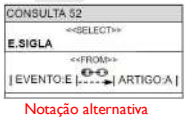
/* CONSULTA 52 - PROJETAR AS SIGLAS DOS EVENTOS SEM ARTIGOS PUBLICADOS – USANDO "NOT EXISTS" */

```
SELECT E.SIGLA
FROM EVENTO E
WHERE  NOT EXISTS
        (SELECT *
         FROM ARTIGO A
         WHERE A.COD = E.COD);
```



ATENÇÃO:

É possível fazer com LEFT JOIN + FK IS NULL, mas não é tão performático quanto o ANTI JOIN.



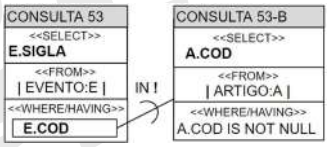
Notação alternativa

DML - Consultando Dados em Tabelas

- **Exemplo** (negação do semi join ou anti join)

/* CONSULTA 53 - PROJETAR AS SIGLAS DOS EVENTOS SEM ARTIGOS PUBLICADOS – USANDO "NOT IN" */

```
SELECT E.SIGLA
FROM EVENTO E
WHERE E.COD  NOT IN
        (SELECT A.COD
         FROM ARTIGO A
         WHERE A.COD IS NOT NULL);
```

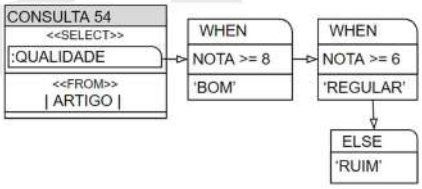


DML - Consultando Dados em Tabelas

- **Exemplo** (avaliação condicional)

-- CONSULTA 54 - PROJETAR A QUALIDADE DO ARTIGO

```
SELECT
   CASE
      WHEN NOTA >= 8 THEN 'BOM'
      WHEN NOTA >= 6 THEN 'REGULAR'
      ELSE 'RUIM'
   END AS QUALIDADE
FROM ARTIGO
```

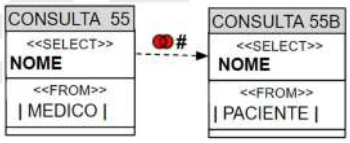


DML - Consultando Dados em Tabelas

- Considere:
  - MEDICO (CODMEDICO, NOME, CRM)
  - PACIENTE (CODPACIENTE, NOME)
- **UNION**

/* CONSULTA 55 - PROJETAR O NOME DE TODAS AS PESSOAS CADASTRADAS NO HOSPITAL */

```
   (SELECT NOME
    FROM MEDICO)
UNION
   (SELECT NOME
    FROM PACIENTE);
```

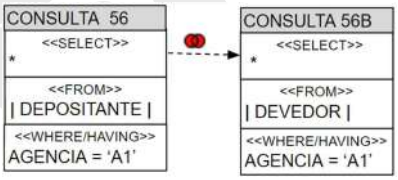


DML - Consultando Dados em Tabelas

- Considere:
  - DEPOSITANTE (CPF, NOME, AGENCIA, CONTA)
  - DEVEDOR (CPF, NOME, AGENCIA, CONTA)
- **UNION**

/* CONSULTA 56 - Projetar todos os clientes da agência A1 com empréstimo ou depósito */

```
   (SELECT *
    FROM DEPOSITANTE
    WHERE AGENCIA = 'A1')
UNION ALL
   (SELECT *
    FROM DEVEDOR
    WHERE AGENCIA = 'A1');
```

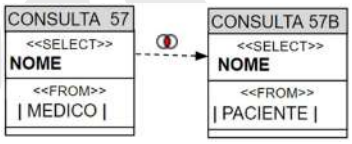


DML - Consultando Dados em Tabelas

- Considere:
  - MEDICO (CODMEDICO, NOME, CRM)
  - PACIENTE (CODPACIENTE, NOME)
- **INTERSECT**

/* CONSULTA 57- PROJETAR O NOME DE TODAS AS PESSOAS QUE SÃO MÉDICOS E PACIENTES AO MESMO TEMPO */

```
   (SELECT NOME
    FROM MEDICO)
INTERSECT
   (SELECT NOME
    FROM PACIENTE);
```

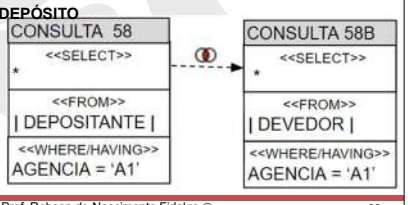


DML - Consultando Dados em Tabelas

- Considere:
  - DEPOSITANTE (CPF, NOME, AGENCIA, CONTA)
  - DEVEDOR (CPF, NOME, AGENCIA, CONTA)
- **INTERSECT**

/* CONSULTA 58 - PROJETAR TODOS OS CLIENTES DA AGÊNCIA A1 COM EMPRÉSTIMO E DEPÓSITO

```
   (SELECT *
    FROM DEPOSITANTE
    WHERE AGENCIA = 'A1')
INTERSECT
   (SELECT *
    FROM DEVEDOR
    WHERE AGENCIA = 'A1');
```

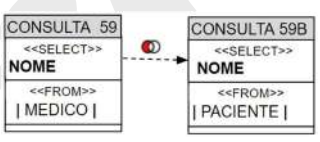


DML - Consultando Dados em Tabelas

- Considere:
  - MEDICO (CODMEDICO, NOME, CRM)
  - PACIENTE (CODPACIENTE, NOME)
- **EXCEPT**

/* CONSULTA 59 - PROJETAR O NOME DE TODAS AS PESSOAS QUE SÃO MÉDICOS E NÃO SÃO PACIENTES */

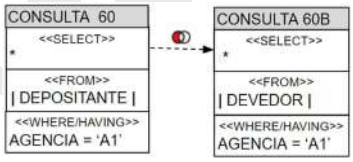
```
   (SELECT NOME
    FROM MEDICO)
EXCEPT
   (SELECT NOME
    FROM PACIENTE);
```



DML - Consultando Dados em Tabelas

- Considere:
  - DEPOSITANTE (CPF, NOME, AGENCIA, CONTA) E
  - DEVEDOR (CPF, NOME, AGENCIA, CONTA).
- **EXCEPT**

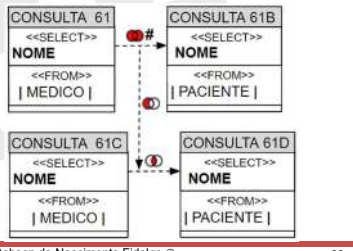
```
                                /*  CONSULTA 60 - PROJETAR TODOS OS CLIENTES
    (SELECT *                       DA AGÊNCIA A1 COM CONTA E SEM EMPRÉSTIMO */
    FROM DEPOSITANTE
    WHERE AGENCIA = 'A1')
EXCEPT
    (SELECT *
    FROM DEVEDOR
    WHERE AGENCIA = 'A1');
```



DML - Consultando Dados em Tabelas

- Considere:
  - MEDICO (CODMEDICO, NOME, CRM) E
  - PACIENTE (CODPACIENTE, NOME);
- **UNION EXCLUSIVA (OU COMPLEMENTO)**

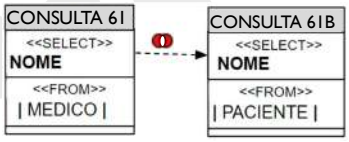
```
    ((SELECT NOME                /* CONSULTA 61 - PROJETAR O NOME DE TODAS AS
    FROM MEDICO)                    PESSOAS QUE APENAS SÃO MÉDICO OU PACIENTE */
  UNION
    (SELECT NOME
    FROM PACIENTE))
EXCEPT
    ((SELECT NOME
    FROM MEDICO)
  INTERSECT
    (SELECT NOME
    FROM PACIENTE));
```



DML - Consultando Dados em Tabelas

- Considere:
  - MEDICO (CodMedico, Nome, CRM) e
  - PACIENTE (CodPaciente, Nome);
- **UNION EXCLUSIVA (OU COMPLEMENTO)**

```
  SELECT NOME
  FROM MEDICO                   /* CONSULTA 61 - PROJETAR O NOME DE TODAS AS
  WHERE NOME NOT IN                PESSOAS QUE APENAS SÃO MÉDICO OU PACIENTE */
    (SELECT NOME
     FROM PACIENTE)
UNION
  SELECT NOME
  FROM MEDICO
  WHERE NOME NOT IN
    (SELECT NOME
    FROM PACIENTE);
```

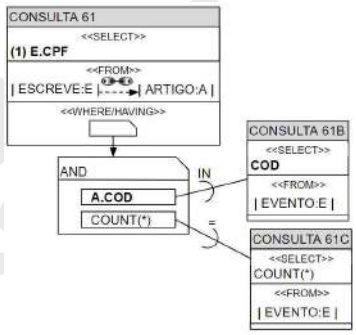


Notação alternativa

DML - Consultando Dados em Tabelas

- **DIVISION**

```
/* PROJETAR OS PROFESSORES QUE
   PUBLICARAM EM TODOS OS EVENTOS */
SELECT E.CPF
FROM ESCREVE E INNER JOIN
      ARTIGO A ON (E.MAT = A.MAT)
WHERE A.COD IN
      (SELECT COD
      FROM EVENTO)
GROUP BY E.CPF
HAVING COUNT(*) =
      (SELECT COUNT(*)
      FROM EVENTO)
```



DML - Consultando Dados em Tabelas

- **DIVISION**

Lembrando: R(X) = R(A) ÷ R(B), onde:
1) B ⊆ A
2) X = A-B
3) R1(A) → Dividendo
4) R2(B) → Divisor
5) R(X) → Quociente

```
/* PROJETAR OS PROFESSORES QUE
   PUBLICARAM EM TODOS OS EVENTOS */
SELECT E.CPF
FROM ESCREVE E INNER JOIN
      ARTIGO A ON (E.MAT = A.MAT)
WHERE A.COD IN
      (SELECT COD
      FROM EVENTO)
GROUP BY E.CPF
HAVING COUNT(*) =
      (SELECT COUNT(*)
      FROM EVENTO)
```

Notação alternativa

APÊNDICE D – QUESTÕES DO EXPERIMENTO 1

TEAM A

TASK 1

```
select p.productname
from suppliers s
   join products p on (p.supplierid = s.supplierid)
where s.region = 'Québec'
   and not exists
   (  select *
      from order_details od
         join orders o on (o.orderid = od.orderid)
         join customers c on (c.customerid = o.customerid)
      where od.productid = p.productid
         and c.region = 'Quebec'
   )
```

What question is answered in the above query:

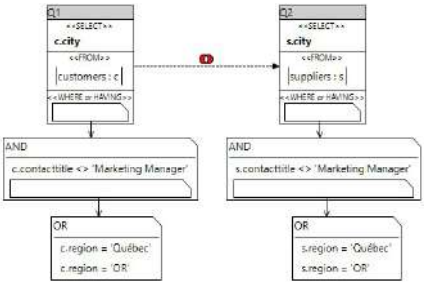
1) What are the product names of suppliers who are outside the Quebec region and have never been ordered by customers in the Quebec region?

2) What are the product names of Quebec region suppliers that have never been ordered by customers outside the Quebec region?

3) What are the product names of suppliers who are outside the Quebec region and have never been ordered by customers who are also outside the Quebec region?

4) What are the product names of Quebec region suppliers that have never been ordered by customers in the same region?

TASK 2



What question is answered in the above query:

1) What are the cities of people who are exclusively customers or suppliers, are not Marketing Manager and live in the Quebec or OR region?

2) What are the cities of people who are customers or suppliers, are not Marketing Manager and live in the Quebec or OR region?

3) What are the people's cities that are customers and suppliers, are they not Marketing Manager and live in the Quebec or OR region?

4) What are the cities of people who are customers, are not suppliers, are not Marketing Manager and live in the Quebec or OR region?

TASK 3

```
Select
   p.productid,
   SUM(od.Quantity * od.UnitPrice) as totalOrdered,
   SUM(od.Quantity * od.UnitPrice) - (
         select avg(od2.Quantity * od2.UnitPrice)
         from Orders o2
            join order_details od2 on (o2.OrderID = od2.OrderID)
            join products p2 on (p2.productid = od2.productid)
         where EXTRACT(YEAR FROM o2.OrderDate) = 2016
            and p2.categoryid = p.categoryid ) as TotalAboveAvg
from Orders o
   join order_details od on (o.OrderID = od.OrderID)
   join products p on (p.productid = od.productid)
where EXTRACT(YEAR FROM o.OrderDate) = 2016
Group by p.productid
Having sum(od.Quantity * od.UnitPrice) > (
         select avg(od2.Quantity * od2.UnitPrice)
         from Orders o2
            join order_details od2 on (o2.OrderID = od2.OrderID)
            join products p2 on (p2.productid = od2.productid)
         where EXTRACT(YEAR FROM o2.OrderDate) = 2016
            and p2.categoryid = p.categoryid )
Order by totalOrdered asc, productid desc;
```

What question is answered in the above query:

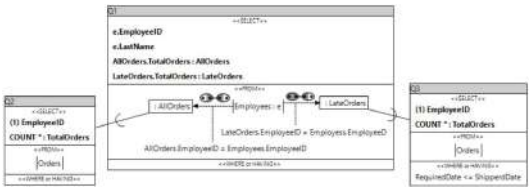
1) In 2016, what were the totals and the above average value of the products that had their orders above the general average of the products? Sort the result ascending, by the total sold, and, in descending order, by the name of the product.

2) In 2016, what were the totals and the above average value of the products that had their orders above the average of the products in the same category? Sort the result, ascending by the total sold, and, in descending order, by the name of the product.

3) In 2016, what were the totals and the value above the general average of the products that had their orders above the average of the products in the same category? Sort the result, ascending by the total sold, and, in descending order, by the name of the product.

4) In 2016, what were the totals and the above average value of the same category products that had their orders above the general average of the products? Sort the result, ascending by the total sold, and, in descending order, by the name of the product.

TASK 4



What question is answered in the above query:

1) What are the vendors' identifiers and last names, as well as the total quantities of their completed orders and their late orders, only for sellers who have at least one backorder?

2) What are the vendors' identifiers and last names, as well as the total quantities of your completed and late orders, even when the seller has not placed any orders?

3) What are the vendors' identifiers and last names, as well as the total quantities of their completed orders and their late orders, even when the seller has no backorders?

4) What are the vendors' identifiers and last names, as well as the total quantities of your completed and late orders, only to sellers who placed orders?

TASK 5

```
Select o.OrderID,
   count(*) as TotalOrderDetails
From Orders o
   Join order_details od on (o.OrderID = od.OrderID)
Where o.ShipCountry = 'Brazil'
   and EXTRACT(YEAR FROM o.OrderDate) IN (2013, 2015, 2017)
   and o.requiredDate < o.shippedDate
Group By o.OrderID
Order By count(*) desc
limit 10 OFFSET 0
```

What question is answered in the above query:

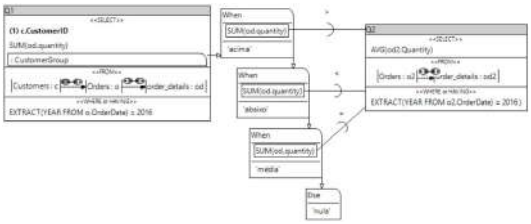
1) What were the 10 orders, made in 2013, 2015, 2017, that were not delayed, were delivered in Brazil and had the products with the highest quantity requested?

2) What were the 10 orders made in 2013, 2015 or 2017 that were delayed, delivered in Brazil and had the largest quantity of requested product items?

3) What were the 10 orders, made in 2013, 2015 or 2017, which were delayed, delivered in Brazil and had the lowest quantity of requested product items?

4) What were the 10 requests, made in 2013, 2015 or 2017, which were not delayed, were delivered in Brazil and had the products with the lowest quantity requested?

TASK 6



What question is answered in the above query:

1) What were the customers, the amount of orders they made in 2016 and if this quantity is above, below, average or null?

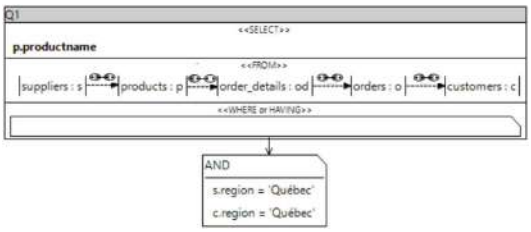
2) What were the customers, the quantities of products they ordered in 2016 and if this quantity is above, below or average?

3) What were the customers, the amount of orders they made in 2016 and if this quantity is above, below or average?

4) What were the customers, the quantities of products they ordered in 2016 and if this quantity is above, below, average or null?

TEAM B

TASK 1



What question is answered in the above query:

1) What are the product names of suppliers who are outside the Quebec region and have never been ordered by customers in the Quebec region?

2) What are the product names of Quebec region suppliers that have never been ordered by customers outside the Quebec region?

3) What are the product names of suppliers who are outside the Quebec region and have never been ordered by customers who are also outside the Quebec region?

4) What are the product names of Quebec region suppliers that have never been ordered by customers in the same region?

TASK 2

```
        (select c.city
        from  customers c
        where c.contacttitle <> 'Marketing Manager'
           and (c.region = 'Québec'
                 or c.region = 'OR')
        union
        select s.city
        from suppliers s
        where s.contacttitle <> 'Marketing Manager'
           and (s.region = 'Québec'
                 or s.region = 'OR' )
        )
except
        (select c.city
        from customers c
        where c.contacttitle <> 'Marketing Manager'
           and (c.region = 'Québec'
              or c.region = 'OR' )
        intersect
        select s.city
        from suppliers s
        where s.contacttitle <> 'Marketing Manager'
           and ( s.region = 'Québec'
              or s.region = 'OR' )
        )
```

What question is answered in the above query:

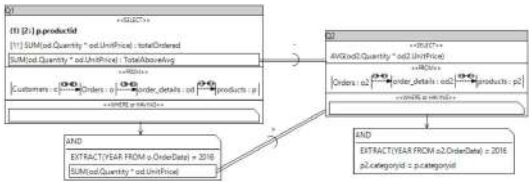
1) What are the cities of people who are exclusively customers or suppliers, are not Marketing Manager and live in the Quebec or OR region?

2) What are the cities of people who are customers or suppliers, are not Marketing Manager and live in the Quebec or OR region?

3) What are the people's cities that are customers and suppliers, are they not Marketing Manager and live in the Quebec or OR region?

4) What are the cities of people who are customers, are not suppliers, are not Marketing Manager and live in the Quebec or OR region?

TASK 3



What question is answered in the above query:

1) In 2016, what were the totals and the above average value of the products that had their orders above the general average of the products? Sort the result ascending, by the total sold, and, in descending order, by the name of the product.

2) In 2016, what were the totals and the above average value of the products that had their orders above the average of the products in the same category? Sort the result, ascending by the total sold, and, in descending order, by the name of the product.

3) In 2016, what were the totals and the value above the general average of the products that had their orders above the average of the products in the same category? Sort the result, ascending by the total sold, and, in descending order, by the name of the product.

4) In 2016, what were the totals and the above average value of the same category products that had their orders above the general average of the products? Sort the result, ascending by the total sold, and, in descending order, by the name of the product.

TASK 4

```
Select e.EmployeeID,
   e.LastName,
   AllOrders.TotalOrders as AllOrders,
   LateOrders.TotalOrders as LateOrders
From Employees e
   Join
     (Select EmployeeID,
           Count(*) as TotalOrders
        From Orders
        Group By EmployeeID
     ) as AllOrders
      on AllOrders.EmployeeID = Employees.EmployeeID
   Left Join
     ( Select EmployeeID,
           Count(*) as TotalOrders
        From Orders
        Where RequiredDate <= ShippedDate
        Group By EmployeeID
     ) as LateOrders
     on LateOrders.EmployeeID = Employees.EmployeeID
```

What question is answered in the above query:

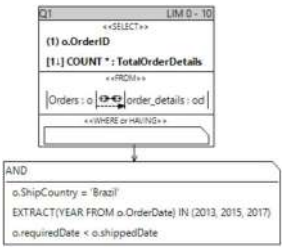
1) What are the vendors' identifiers and last names, as well as the total quantities of their completed orders and their late orders, only for sellers who have at least one backorder?

2) What are the vendors' identifiers and last names, as well as the total quantities of your completed and late orders, even when the seller has not placed any orders?

3) What are the vendors' identifiers and last names, as well as the total quantities of their completed orders and their late orders, even when the seller has no backorders?

4) What are the vendors' identifiers and last names, as well as the total quantities of your completed and late orders, only to sellers who placed orders?

TASK 5



What question is answered in the above query:

1) What were the 10 orders, made in 2013, 2015, 2017, that were not delayed, were delivered in Brazil and had the products with the highest quantity requested?

2) What were the 10 orders made in 2013, 2015 or 2017 that were delayed, delivered in Brazil and had the largest quantity of requested product items?

3) What were the 10 orders, made in 2013, 2015 or 2017, which were delayed, delivered in Brazil and had the lowest quantity of requested product items?

4) What were the 10 requests, made in 2013, 2015 or 2017, which were not delayed, were delivered in Brazil and had the products with the lowest quantity requested?

TASK 6

```
Select c.CustomerID,
   SUM(od.quantity),
   Case
      when SUM(od.quantity) > (
         select avg(od2.Quantity)
         from Orders o2
            join order_details od2 on (o2.OrderID = od2.OrderID)
         where EXTRACT(YEAR FROM o2.OrderDate) = 2016 )
      then 'acima'
      when SUM(od.quantity) < (
         select avg(od2.Quantity)
         from Orders o2
            join order_details od2 on (o2.OrderID = od2.OrderID)
         where EXTRACT(YEAR FROM o2.OrderDate) = 2016 )
      then 'abaixo'
      when SUM(od.quantity) = (
         select avg(od2.Quantity)
         from Orders o2
            join order_details od2 on (o2.OrderID = od2.OrderID)
         where EXTRACT(YEAR FROM o2.OrderDate) = 2016)
      then 'média'
      else 'nula' end as CustomerGroup
from Customers c
   left join Orders o on o.CustomerID = c.CustomerID
   inner join order_details od on o.OrderID = od.OrderID
where EXTRACT(YEAR FROM o.OrderDate) = 2016
Group by c.CustomerID
```

What question is answered in the above query:

1) What were the customers, the amount of orders they made in 2016 and if this quantity is above, below, average or null?

2) What were the customers, the quantities of products they ordered in 2016 and if this quantity is above, below or average?

3) What were the customers, the amount of orders they made in 2016 and if this quantity is above, below or average?

4) What were the customers, the quantities of products they ordered in 2016 and if this quantity is above, below, average or null?

answers

1) 4
2) 1
3) 2
4) 3
5) 2
6) 4

APÊNDICE E – QUESTÕES DO EXPERIMENTO 2

TEAM A

TASK 1

```
select p.productname
from suppliers s
   join products p on (p.supplierid = s.supplierid)
where s.region = 'Québec'
   and not exists
   (  select *
      from order_details od
         join orders o on (o.orderid = od.orderid)
         join customers c on (c.customerid = o.customerid)
      where od.productid = p.productid
         and c.region = 'Quebec'
   )
```

What question is answered in the above query:

1) What are the product names of suppliers who are outside the Quebec region and have never been ordered by customers in the Quebec region?

2) What are the product names of Quebec region suppliers that have never been ordered by customers outside the Quebec region?

3) What are the product names of suppliers who are outside the Quebec region and have never been ordered by customers who are also outside the Quebec region?

4) What are the product names of Quebec region suppliers that have never been ordered by customers in the same region?

TASK 2

```
      (select c.city
      from  customers c
      where c.contacttitle <> 'Marketing Manager'
         and (c.region = 'Québec'
               or c.region = 'OR')
      union
      select s.city
      from suppliers s
      where s.contacttitle <> 'Marketing Manager'
         and (s.region = 'Québec'
               or s.region = 'OR' )
   )
except
      (select c.city
      from customers c
      where c.contacttitle <> 'Marketing Manager'
         and (c.region = 'Québec'
            or c.region = 'OR' )
      intersect
      select s.city
      from suppliers s
      where s.contacttitle <> 'Marketing Manager'
         and ( s.region = 'Québec'
            or s.region = 'OR' )
   )
```

What question is answered in the above query:

1) What are the cities of people who are exclusively customers or suppliers, are not Marketing Manager and live in the Quebec or OR region?

2) What are the cities of people who are customers or suppliers, are not Marketing Manager and live in the Quebec or OR region?

3) What are the people's cities that are customers and suppliers, are they not Marketing Manager and live in the Quebec or OR region?

4) What are the cities of people who are customers, are not suppliers, are not Marketing Manager and live in the Quebec or OR region?

TASK 3

```
Select
   p.productid,
   SUM(od.Quantity * od.UnitPrice) as totalOrdered,
   $UM(od.Quantity * od.UnitPrice) - (
         select avg(od2.Quantity * od2.UnitPrice)
         from Orders o2
            join order_details od2 on (o2.OrderID = od2.OrderID)
            join products p2 on (p2.productid = od2.productid)
         where EXTRACT(YEAR FROM o2.OrderDate) = 2016
            and p2.categoryid = p.categoryid ) as TotalAboveAvg
from Orders o
   join order_details od on (o.OrderID = od.OrderID)
   join products p on (p.productid = od.productid)
Where EXTRACT(YEAR FROM o.OrderDate) = 2016
Group by p.productid
Having sum(od.Quantity * od.UnitPrice) > (
         select avg(od2.Quantity * od2.UnitPrice)
         from Orders o2
            join order_details od2 on (o2.OrderID = od2.OrderID)
            join products p2 on (p2.productid = od2.productid)
         where EXTRACT(YEAR FROM o2.OrderDate) = 2016
            and p2.categoryid = p.categoryid )
Order by totalOrdered asc, productid desc;
```

What question is answered in the above query:

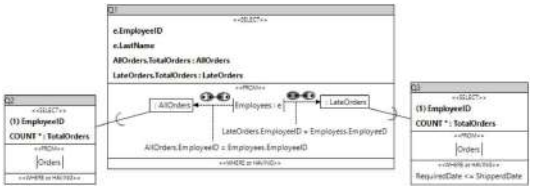
1) In 2016, what were the totals and the above average value of the products that had their orders above the general average of the products? Sort the result ascending, by the total sold, and, in descending order, by the name of the product.

2) In 2016, what were the totals and the above average value of the products that had their orders above the average of the products in the same category? Sort the result, ascending by the total sold, and, in descending order, by the name of the product.

3) In 2016, what were the totals and the value above the general average of the products that had their orders above the average of the products in the same category? Sort the result, ascending by the total sold, and, in descending order, by the name of the product.

4) In 2016, what were the totals and the above average value of the same category products that had their orders above the general average of the products? Sort the result, ascending by the total sold, and, in descending order, by the name of the product.

TASK 4



What question is answered in the above query:

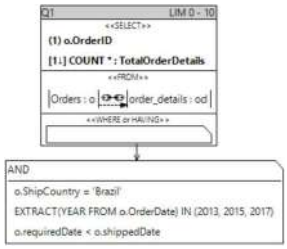
1) What are the vendors' identifiers and last names, as well as the total quantities of their completed orders and their late orders, only for sellers who have at least one backorder?

2) What are the vendors' identifiers and last names, as well as the total quantities of your completed and late orders, even when the seller has not placed any orders?

3) What are the vendors' identifiers and last names, as well as the total quantities of their completed orders and their late orders, even when the seller has no backorders?

4) What are the vendors' identifiers and last names, as well as the total quantities of your completed and late orders, only to sellers who placed orders?

TASK 5



What question is answered in the above query:

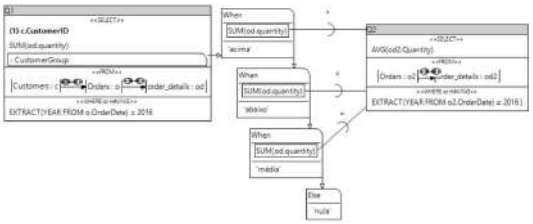
1) What were the 10 orders, made in 2013, 2015, 2017, that were not delayed, were delivered in Brazil and had the products with the highest quantity requested?

2) What were the 10 orders made in 2013, 2015 or 2017 that were delayed, delivered in Brazil and had the largest quantity of requested product items?

3) What were the 10 orders, made in 2013, 2015 or 2017, which were delayed, delivered in Brazil and had the lowest quantity of requested product items?

4) What were the 10 requests, made in 2013, 2015 or 2017, which were not delayed, were delivered in Brazil and had the products with the lowest quantity requested?

TASK 6



What question is answered in the above query:

1) What were the customers, the amount of orders they made in 2016 and if this quantity is above, below, average or null?

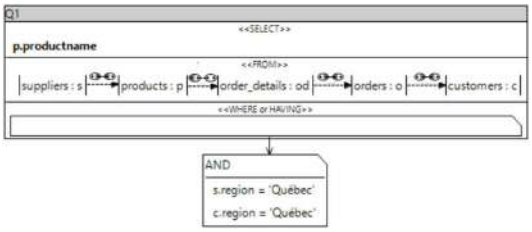
2) What were the customers, the quantities of products they ordered in 2016 and if this quantity is above, below or average?

3) What were the customers, the amount of orders they made in 2016 and if this quantity is above, below or average?

4) What were the customers, the quantities of products they ordered in 2016 and if this quantity is above, below, average or null?

TEAM B

TASK 1



What question is answered in the above query:

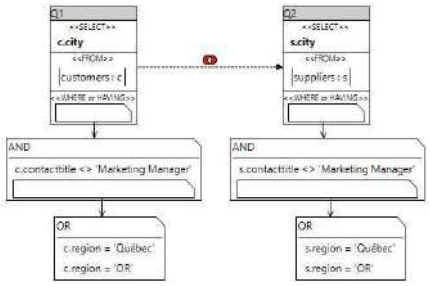
1) What are the product names of suppliers who are outside the Quebec region and have never been ordered by customers in the Quebec region?

2) What are the product names of Quebec region suppliers that have never been ordered by customers outside the Quebec region?

3) What are the product names of suppliers who are outside the Quebec region and have never been ordered by customers who are also outside the Quebec region?

4) What are the product names of Quebec region suppliers that have never been ordered by customers in the same region?

TASK 2



What question is answered in the above query:

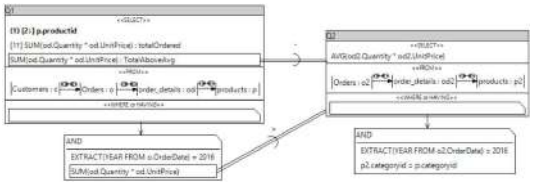
1) What are the cities of people who are exclusively customers or suppliers, are not Marketing Manager and live in the Quebec or OR region?

2) What are the cities of people who are customers or suppliers, are not Marketing Manager and live in the Quebec or OR region?

3) What are the people's cities that are customers and suppliers, are they not Marketing Manager and live in the Quebec or OR region?

4) What are the cities of people who are customers, are not suppliers, are not Marketing Manager and live in the Quebec or OR region?

TASK 3



What question is answered in the above query:

1) In 2016, what were the totals and the above average value of the products that had their orders above the general average of the products? Sort the result ascending, by the total sold, and, in descending order, by the name of the product.

2) In 2016, what were the totals and the above average value of the products that had their orders above the average of the products in the same category? Sort the result, ascending by the total sold, and, in descending order, by the name of the product.

3) In 2016, what were the totals and the value above the general average of the products that had their orders above the average of the products in the same category? Sort the result, ascending by the total sold, and, in descending order, by the name of the product.

4) In 2016, what were the totals and the above average value of the same category products that had their orders above the general average of the products? Sort the result, ascending by the total sold, and, in descending order, by the name of the product.

TASK 4

TASK 5

```
Select e.EmployeeID,
    e.LastName,
    AllOrders.TotalOrders as AllOrders,
    LateOrders.TotalOrders as LateOrders
From Employees e
    Join
      (Select EmployeeID,
            Count(*) as TotalOrders
        From Orders
        Group By EmployeeID
      ) as AllOrders
      on AllOrders.EmployeeID = Employees.EmployeeID
    Left Join
      ( Select EmployeeID,
            Count(*) as TotalOrders
        From Orders
        Where RequiredDate <= ShippedDate
        Group By EmployeeID
      ) as LateOrders
      on LateOrders.EmployeeID = Employees.EmployeeID
```

What question is answered in the above query:

1) What are the vendors' identifiers and last names, as well as the total quantities of their completed orders and their late orders, only for sellers who have at least one backorder?

2) What are the vendors' identifiers and last names, as well as the total quantities of your completed and late orders, even when the seller has not placed any orders?

3) What are the vendors' identifiers and last names, as well as the total quantities of their completed orders and their late orders, even when the seller has no backorders?

4) What are the vendors' identifiers and last names, as well as the total quantities of your completed and late orders, only to sellers who placed orders?

```
Select o.OrderID,
   count(*) as TotalOrderDetails
From Orders o
   Join order_details od on (o.OrderID = od.OrderID)
Where o.ShipCountry = 'Brazil'
   and EXTRACT(YEAR FROM o.OrderDate) IN (2013, 2015, 2017)
   and o.requiredDate < o.shippedDate
Group By o.OrderID
Order By count(*) desc
limit 10 OFFSET 0
```

What question is answered in the above query:

1) What were the 10 orders, made in 2013, 2015, 2017, that were not delayed, were delivered in Brazil and had the products with the highest quantity requested?

2) What were the 10 orders made in 2013, 2015 or 2017 that were delayed, delivered in Brazil and had the largest quantity of requested product items?

3) What were the 10 orders, made in 2013, 2015 or 2017, which were delayed, delivered in Brazil and had the lowest quantity of requested product items?

4) What were the 10 requests, made in 2013, 2015 or 2017, which were not delayed, were delivered in Brazil and had the products with the lowest quantity requested?

TASK 6

```
Select c.CustomerID,
   SUM(od.quantity),
   case
      when SUM(od.quantity) > (
         select avg(od2.Quantity)
         from Orders o2
            join order_details od2 on (o2.OrderID = od2.OrderID)
         where EXTRACT(YEAR FROM o2.OrderDate) = 2016 )
      then 'acima'
      when SUM(od.quantity) < (
         select avg(od2.Quantity)
         from Orders o2
            join order_details od2 on (o2.OrderID = od2.OrderID)
         where EXTRACT(YEAR FROM o2.OrderDate) = 2016 )
      then 'abaixo'
      when SUM(od.quantity) = (
         select avg(od2.Quantity)
         from Orders o2
            join order_details od2 on (o2.OrderID = od2.OrderID)
         where EXTRACT(YEAR FROM o2.OrderDate) = 2016)
      then 'média'
      else 'nula' end as CustomerGroup
From Customers c
   left join Orders o on o.CustomerID = c.CustomerID
   inner join order_details od on o.OrderID = od.OrderID
Where EXTRACT(YEAR FROM o.OrderDate) = 2016
Group by c.CustomerID
```

What question is answered in the above query:

1) What were the customers, the amount of orders they made in 2016 and if this quantity is above, below, average or null?

2) What were the customers, the quantities of products they ordered in 2016 and if this quantity is above, below or average?

3) What were the customers, the amount of orders they made in 2016 and if this quantity is above, below or average?

4) What were the customers, the quantities of products they ordered in 2016 and if this quantity is above, below, average or null?

answers

1) 4
2) 1
3) 2
4) 3
5) 2
6) 4